

Lukasiewicz μ -calculus

Matteo Mio¹ and Alex Simpson²

¹CNRS and École Normale Supérieure de Lyon, France

²Faculty of Mathematics and Physics, University of Ljubljana, Slovenia

Abstract

The paper explores properties of the *Lukasiewicz μ -calculus*, or $L\mu$ for short, an extension of Lukasiewicz logic with scalar multiplication and least and greatest fixed-point operators (for monotone formulas). We observe that $L\mu$ terms, with n variables, define monotone piecewise linear functions from $[0, 1]^n$ to $[0, 1]$. Two effective procedures for calculating the output of $L\mu$ terms on rational inputs are presented. We then consider the Lukasiewicz *modal μ -calculus*, which is obtained by adding box and diamond modalities to $L\mu$. Alternatively, it can be viewed as a generalization of Kozen's modal μ -calculus adapted to probabilistic nondeterministic transition systems (PNTS's). We show how properties expressible in the well-known logic **PCTL** can be encoded as Lukasiewicz modal μ -calculus formulas. We also show that the algorithms for computing values of Lukasiewicz μ -calculus terms provide automatic (albeit impractical) methods for verifying Lukasiewicz modal μ -calculus properties of finite rational PNTS's.

1 Introduction

This paper has two distinct motivations. The first is purely logical. The paper investigates certain properties of a very natural extension of Lukasiewicz many-valued logic, obtained by adding scalar multiplication and least and greatest fixed-point operators for monotone formulas. Whereas, an extension with scalar multiplication has previously been investigated (for example in [31]), the addition of monotone least and greatest fixed-points has not been systematically studied before. (See Section 2, however, for discussion of related work by Spada [34].)

Our second motivation is more practical. We propose Lukasiewicz logic with fixed points as a useful logic for the specification and verification of systems combining nondeterminism and probabilistic behaviour. We now summarize the basis of this perspective.

Among logics for expressing properties of nondeterministic (including concurrent) processes, represented as transition systems, Kozen's modal μ -calculus [20] plays a fundamental role. It subsumes other temporal logics of processes, such as **LTL**, **CTL** and **CTL***. It does not distinguish bisimilar processes, but separates (finite) non-bisimilar ones. More generally, by a remarkable result of Janin and Walukiewicz [19], it is exactly as expressive as the bisimulation-invariant fragment of monadic second-order logic. Furthermore, there is an intimate connection with parity games, which offers an intuitive reading of fixed-points, and underpins the existing technology for model-checking μ -calculus properties.

For many purposes, it is useful to add probability to the computational model, leading to probabilistic nondeterministic transition systems, cf. [32]. Among the different approaches that have been followed to developing analogues of the modal μ -calculus in this setting, the most significant is that introduced independently by Huth and Kwiatkowska [17] and by Morgan and McIver [29], under which a *quantitative* interpretation is given, with formulas denoting values in $[0, 1]$. This quantitative setting permits several variations. In particular, three different quantitative extensions of conjunction from booleans to $[0, 1]$ (with 0 as false and 1 as true) arise naturally [17]: minimum, $\min(x, y)$; multiplication, xy ; and the strong conjunction (a.k.a. Lukasiewicz t-norm) from Lukasiewicz fuzzy logic, $\max(x + y - 1, 0)$. In each case, there is a dual operator giving a corresponding extension of disjunction: maximum, $\max(x, y)$; comultiplication, $x + y - xy$; and Lukasiewicz strong disjunction, $\min(x + y, 1)$. The choice of min and max for conjunction and disjunction is particularly natural, since the corresponding quantitative modal μ -calculus, called $qL\mu$ in [23], has an interpretation in terms of two-player *stochastic* parity games,

which extends the usual parity-game interpretation of the ordinary modal μ -calculus. This allows the real number denoted by a formula to be understood as the *value* of the associated game [23, 26].

The present paper contributes to a programme of ongoing research, one of whose overall aims is to investigate the extent to which quantitative modal μ -calculi play as fundamental a role in the probabilistic setting as that of Kozen’s modal μ -calculus in the nondeterministic setting. The logic $\text{qL}\mu$, with \min/\max as conjunction/disjunction, is insufficiently expressive. For example, it cannot encode the standard probabilistic temporal logic **PCTL** of [2]. Nevertheless, richer calculi can be obtained by augmenting $\text{qL}\mu$ with the other alternatives for conjunction/disjunction, to be used in combination with \max and \min . Such extensions were investigated by the first author in [27, 25], where the game-theoretic interpretation was generalized to accommodate the new operations.

In this paper, we study the power of least and greatest fixed points in the presence of both the weak conjunction and disjunction (\min and \max —written as \sqcap and \sqcup) as well as the Łukasiewicz strong connectives (written as \odot and \oplus) in combination. In addition, as already mentioned we include a basic operation for multiplying the value of a formula by a rational constant in $[0, 1]$. We call the resulting logic *Łukasiewicz μ -calculus*. We consider both a basic version with just the operations mentioned above, and a modal version with added \Box and \Diamond modalities.

As a first contribution of this paper, we consider the problem of computing the value of a (non-modal) Łukasiewicz μ -calculus formula. We describe two approaches to this. The first is indirectly obtained by a reduction to the decision problem of the first order theory of rational linear arithmetic, which admits quantifier elimination. As a consequence of this property, Łukasiewicz μ -calculus terms always denote piecewise linear (generally discontinuous) functions. The second algorithm adapts the approximation-based approach to nested fixed-point calculation to our quantitative setting.

As a second contribution, we show that the the well-known probabilistic temporal logic **PCTL** [2] can be encoded in the modal version of Łukasiewicz μ -calculus. A similar translation was originally given in the first author’s PhD thesis [25]. In this journal presentation of the result, we streamline the treatment slightly by removing the connectives of *independent product* and its dual from the target μ -calculus. The translation provides some evidence that Łukasiewicz modal μ -calculus provides an expressive logic for specifying properties of systems combining probabilistic choice and nondeterminism.

Finally, we show how the computation of the value of a Łukasiewicz modal μ -calculus formula in a finite probabilistic transition system can be reduced to the problem of computing the value of a formula in the modality-free calculus. This, in theory, allows the algorithms we consider for computing such values to be brought to bear on the quantitative model-checking problem for Łukasiewicz modal μ -calculus. Also, again in theory, it subsumes the problem of model-checking **PCTL**. However, whereas the model-checking problem for **PCTL** is known to be polynomial-time solvable in the size of formula and model [1, §10.6.2], the algorithms considered in the present paper have abysmal complexity, at least on paper. To what extent this can be improved is an interesting topic for future work.

In any case, the purpose of the present paper is not to provide a practical algorithmic framework. Rather, it is to explore something of the richness of the combination of Łukasiewicz logic with fixed points, with the idea that this provides a conceptually clean basis for the specification of probabilistic systems, and one that is worthy of further investigation as a potential tool for verification.

2 Łukasiewicz μ -calculus

This paper concerns Łukasiewicz many-valued (“fuzzy”) logic, see, e.g., [16, 30]. The main purpose is to study fixed-point extensions of Łukasiewicz logic. As discussed in Section 1, our motivation for considering such extensions comes from an interest in specification logics for probabilistic systems. For this reason, we consider only the standard interpretation of the connectives in the interval $[0, 1]$, rather than, say, in arbitrary MV-algebras [30]. In this context, it is natural also to extend formulas with a construct for “scalar multiplication” by a real r . (Such an extension with scalar multiplication is studied in [31], see also [14, 15], where it is shown to be the logical counterpart of so-called Riesz MV-algebras.)

This leads to a syntax of formulas generated by the following grammar (we do not try to be minimal),

$$t ::= x \mid \underline{0} \mid \underline{1} \mid r t \mid t \sqcup t \mid t \sqcap t \mid t \oplus t \mid t \odot t \mid \neg t ,$$

where x ranges over a countably infinite set of variables, and r ranges over reals in $[0, 1]$. Henceforth, we call the t specified by the grammar above *terms* rather than formulas.

We write $t(x_1, \dots, x_n)$ to mean that all variables of t are contained in $\{x_1, \dots, x_n\}$. Such a term is interpreted as a function of type $[0, 1]^n \rightarrow [0, 1]$, by inductively defining the *value* $t(\vec{r})$ of $t(x_1, \dots, x_n)$ applied to a vector $(r_1, \dots, r_n) \in [0, 1]^n$ as follows:

$$\begin{array}{ll}
x_i(\vec{r}) = r_i & \underline{0}(\vec{r}) = 0 \quad \underline{1}(\vec{r}) = 1 & \text{variables and constants} \\
(qt)(\vec{r}) = q \cdot t(\vec{r}) & & \text{scalar multiplication} \\
(t_1 \sqcup t_2)(\vec{r}) = \max(t_1(\vec{r}), t_2(\vec{r})) & & \text{Łukasiewicz weak disjunction} \\
(t_1 \sqcap t_2)(\vec{r}) = \min(t_1(\vec{r}), t_2(\vec{r})) & & \text{Łukasiewicz weak conjunction} \\
(t_1 \oplus t_2)(\vec{r}) = \min(t_1(\vec{r}) + t_2(\vec{r}), 1) & & \text{Łukasiewicz strong disjunction} \\
(t_1 \odot t_2)(\vec{r}) = \max(t_1(\vec{r}) + t_2(\vec{r}) - 1, 0) & & \text{Łukasiewicz strong conjunction} \\
(\neg t)(\vec{r}) = 1 - t(\vec{r}) & & \text{negation}
\end{array}$$

One way of extending Łukasiewicz logic with least and greatest fixed points has previously been considered in [34]. This is based on the observation that, since any term $t(x_1, \dots, x_n, y)$ denotes a continuous function, the existence of a solution y to the equation $y = t(x_1 \dots x_n, y)$ is guaranteed by the Brouwer fixed-point theorem. Because the space of all solutions is closed, it is thus possible to define the semantics of a least-fixed-point expression $\mu y.t(x_1 \dots x_n, y)$, with $t(x_1 \dots x_n, y)$ an arbitrary Łukasiewicz term, as the smallest solution; and similarly for greatest-fixed-point expressions $\nu y.t(x_1 \dots x_n, y)$. For example the term $\mu x.\neg x$ denotes, under this approach, the rational number $\frac{1}{2}$ (as does $\nu x.\neg x$). However, solutions of fixed-point expressions need not be themselves continuous. For example, $\mu y.(x \oplus y)$ denotes (see Proposition 2.2 below) the discontinuous function

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

As a consequence, the term $\mu x.\neg(\mu y.(x \oplus y))$ cannot be given meaning because the equation $x = 1 - f(x)$ does not have solutions. For this reason, the fixed-point extension of Łukasiewicz logic of [34] does not admit nesting of fixed-point operators.

In this paper we follow an alternative approach to the assignment of values to fixed-point expressions, which allows unrestricted nesting of fixed-point operators. For this, we base our treatment of fixed points on the order-theoretic Knaster-Tarski fixed-point theorem, rather than on the topological Brouwer fixed-point theorem. This approach has been very productive in the development of μ -calculi, see, e.g., [20, 35]. As we shall show, it also leads to a rich and interesting theory in the context of Łukasiewicz logic.

As is necessary for the application of the Knaster-Tarski fixed-point theorem, we consider fixed points of monotone functions only. This is implemented syntactically by considering only terms without negation. The syntax of *Łukasiewicz μ -terms* (or just μ -terms for short) is thus specified by the grammar:

$$t ::= x \mid \underline{0} \mid \underline{1} \mid rt \mid t \sqcup t \mid t \sqcap t \mid t \oplus t \mid t \odot t \mid \mu x.t \mid \nu x.t$$

where, as expected, the μ and ν operators bind their variables. We write $t(x_1 \dots x_n)$ to specify that the *free variables* occurring in t are contained in $\{x_1, \dots, x_n\}$. We say that a μ -term t is *rational* if all scalars r appearing in scalar multiplications in t are rational numbers.

The *value* $t(\vec{r})$ of a μ -term $t(x_1, \dots, x_n)$ applied to a vector $(r_1, \dots, r_n) \in [0, 1]^n$ is specified by extending the definition of value of Łukasiewicz terms as follows:

$$\begin{aligned}
(\mu y.t(x_1 \dots x_n, y))(\vec{r}) &= \text{lfp}(r' \mapsto t(\vec{r}, r')) \\
(\nu y.t(x_1 \dots x_n, y))(\vec{r}) &= \text{gfp}(r' \mapsto t(\vec{r}, r'))
\end{aligned}$$

where lfp and gfp are the operators returning the least and greatest fixed point respectively. These fixed points exist, because $t(x_1, \dots, x_n)$ always defines a monotone function from $[0, 1]^n$ to $[0, 1]$ (by routine induction on t). Note that the values of $\mu x.x$ and $\nu x.x$ are 0 and 1 respectively. Thus the constants $\underline{0}$ and $\underline{1}$ of Łukasiewicz logic are derivable as μ -terms. Given a real $r \in [0, 1]$, we write \underline{r} for the formula $r \underline{1}$ whose value is r .

As is customary in fixed-point logics, we have presented terms in positive form, i.e., without negations. However, an operation $\text{negate}(t)$ can be defined on terms by replacing every connective with its dual, replacing $(q\ t)$ with $(q\ \text{negate}(t)) \oplus \underline{1 - q}$, and using the definitions $\text{negate}(\mu x. t) = \nu x. \text{negate}(t)$ and $\text{negate}(\nu x. t) = \mu x. \text{negate}(t)$ for fixed points. It is simple to verify that, for a term $t(x_1, \dots, x_n)$, it holds that $\text{negate}(t)(\vec{r}) = 1 - t(1 - \vec{r})$, for all $\vec{r} \in [0, 1]^n$. This defines negation for *closed* terms t , since for such terms the equality $\text{negate}(t) = 1 - t$ holds between values.

The next definition introduces a few useful macro formulas.

Definition 2.1. For every μ -term t define:

$$\mathbb{P}_{>0}t = \mu y. (y \oplus t) \quad \mathbb{P}_{=1}t = \nu y. (y \odot t) \quad \mathbb{P}_{>r}t = \mathbb{P}_{>0}(t \odot \underline{1 - r}) \quad \mathbb{P}_{\geq r}t = \mathbb{P}_{=1}(t \oplus \underline{1 - r})$$

where $r \in (0, 1)$ and the variable y does not appear in t . We write $\mathbb{P}_{\times r}t$, for $r \in [0, 1]$, to denote one of the four cases. The derived operators $\mathbb{P}_{\times r}$ are called *threshold modalities*.

Proposition 2.2. Given a μ -term $t(x_1 \dots x_n)$ the following equality holds:

$$(\mathbb{P}_{\times r}t)(\vec{r}) = \begin{cases} 1 & \text{if } t(\vec{r}) \times r \\ 0 & \text{otherwise} \end{cases}$$

Therefore μ -terms generally represent discontinuous functions.

The main goal of this section is to establish the following fundamental properties of the Łukasiewicz μ -calculus. First, any μ -term $t(x_1, \dots, x_n)$ defines a piecewise linear function in $[0, 1]^n \rightarrow [0, 1]$. Second, if the term is rational, then so are the linear pieces; by which we mean that the coefficients and constants in the linear expression for each piece are rational, as are those in the linear expressions specifying the constraints defining the domain of the piece. In particular, if t is a closed rational μ -term, then its value is a rational number. Finally, we show that this rational number is computable from t .

In this section, we formalise and prove these results via a simple reduction to the first-order theory of linear arithmetic. After this, in Section 3, we provide an alternative approach based on a direct iterative algorithm for computing least and greatest fixed points of monotone piecewise linear functions.

A *linear expression* in variables x_1, \dots, x_n is an expression

$$q_1x_1 + \dots + q_nx_n + q$$

where q_1, \dots, q_n, q are real numbers. We say that the linear expression is *rational* when all of q_1, \dots, q_n, q are rational numbers. (The choice of the letter q reflects on the fact that our primary interest will be in the rational case.) We write $e(x_1, \dots, x_n)$ if e is a linear expression in x_1, \dots, x_n , in which case, given real numbers r_1, \dots, r_n , we write $e(\vec{r})$ for the value of the expression when the variables \vec{x} take values \vec{r} . We also make use of the closure of linear expressions under substitution: given $e(x_1, \dots, x_n)$ and $e_1(y_1, \dots, y_m), \dots, e_n(y_1, \dots, y_m)$, we write $e(e_1, \dots, e_n)$ for the evident substituted expression in variables y_1, \dots, y_m (which is defined formally by multiplying out and adding coefficients).

The first-order theory of *linear arithmetic* has linear expressions as terms, and strict and non-strict inequalities between linear expressions,

$$e_1 < e_2 \quad e_1 \leq e_2, \tag{2}$$

as atomic formulas. Equality can be expressed as the conjunction of two non-strict inequalities and the negation of an atomic formula can itself be expressed as an atomic formula. The truth of a first-order formula is given via its interpretation in the reals. The theory of *rational* linear arithmetic is defined identically, but with terms restricted to rational linear expressions.

Both linear arithmetic and rational linear arithmetic enjoy quantifier elimination; see, e.g., [11]. As a consequence, they are model complete. Thus, in the case of rational linear arithmetic, the rational numbers form a model, and the inclusion of the rationals into the reals is an elementary embedding.

Proposition 2.3. For every Łukasiewicz μ -term $t(x_1, \dots, x_n)$, its graph

$$\{(\vec{x}, y) \in [0, 1]^{n+1} \mid t(\vec{x}) = y\}$$

is definable by a formula $F_t(x_1, \dots, x_n, y)$ in the first-order theory of linear arithmetic. In the case that t is rational, it holds that F_t is a rational formula and F_t is computable from t . Furthermore, if u is the length of the μ -term t and v is the number of fixed points constructors in t , the length of F_t is bounded by $2^v u c$, for some constant c .

Proof. The proof is a straightforward induction on the structure of t . We consider two cases, in order to illustrate the simple manipulations used in the construction of F_t .

If t is $t_1 \oplus t_2$ then $F_t(\vec{x}, y)$ is the formula

$$\exists z_1, z_2. F_{t_1}(\vec{x}, z_1) \wedge F_{t_2}(\vec{x}, z_2) \wedge ((z_1 + z_2 \leq 1 \wedge y = z_1 + z_2) \vee (1 \leq z_1 + z_2 \wedge y = 1))$$

If l_1, l_2 are the lengths of the formulas F_{t_1}, F_{t_2} respectively, then the length of $F_{t_1 \oplus t_2}$ is $l \leq l_1 + l_2 + c$.

If t is $\mu x_{n+1}. t'$ then F_t is the formula

$$F_{t'}(x_1, \dots, x_n, y, y) \wedge \forall z. F_{t'}(x_1, \dots, x_n, z, z) \rightarrow y \leq z .$$

If l' is the length of $F_{t'}$ then the length of $F_{\mu x_{n+1}. t'}$ is $l \leq 2l' + c$.

□

Proposition 2.3 provides the following method of computing the value $t(\vec{q})$ of rational Łukasiewicz μ -calculus term $t(x_1, \dots, x_n)$ at a rational vector $(q_1, \dots, q_n) \in [0, 1]^n$. First construct $F_t(x_1, \dots, x_n, y)$. Next, perform quantifier elimination to obtain an equivalent quantifier-free formula $G_t(x_1, \dots, x_n, y)$, and consider its instantiation $G_t(q_1, \dots, q_n, y)$ at \vec{q} . (Alternatively, obtain an equivalent formula $G_t^{\vec{q}}(y)$ by performing quantifier elimination on $F_t(q_1, \dots, q_n, y)$.) By performing obvious simplifications of atomic formulas in one variable, $G_t(q_1, \dots, q_n, y)$ reduces to a boolean combination of inequalities each having one of the following forms

$$y \leq q \quad y < q \quad y \geq q \quad y > q .$$

By the correctness of G_t there must be a unique rational satisfying the Boolean combination of constraints, and this can be extracted in a straightforward way from $G_t(q_1, \dots, q_n, y)$. The quantifier-elimination procedure of [11], when given a formula of length l as input produces a formula of length at most 2^{dl} as output and takes time at most $2^{d'l}$, for some constants d and d' . Thus the length of the formula $G_t(x_1, \dots, x_n, y)$ is bounded by $O(2^{2^v u})$, and the computation time for $t(\vec{q})$ is $O(2^{2^{2^v u}})$, using a unit cost model for rational arithmetic.

Better bounds can be obtained using the decision procedure for linear arithmetic of [4] which is based on automata theoretic methods. Given the formula $F_t(q_1, \dots, q_n, y)$ of length l , one can construct in time 2^{dl} , for some constant d , a deterministic Büchi automaton \mathcal{A} recognizing the set of $(\omega$ -words which are codings of) real numbers r satisfying $F_t(q_1, \dots, q_n, r)$. Once again, by the correctness of F_t , a unique rational number q satisfies the formula $F_t(q_1, \dots, q_n, y)$. The code of q can efficiently be extracted from \mathcal{A} . Thus this procedure computes the value of $t(\vec{q})$ in time bounded by $O(2^{2^v u})$.

Remark 2.4. The practical efficiency of the procedure can be improved by a more careful translation from μ -terms to linear arithmetic (for example, solving adjacent fixed points of the same kind simultaneously to avoid unnecessary quantifier alternation). Nevertheless, even with such improvements, the theoretical bounds discussed above are not reduced. It might be possible to improve the doubly exponential bound to a single exponential one by developing an efficient direct translation of μ -terms into the corresponding Büchi automaton \mathcal{A} (e.g., exploiting the fact that the formula $F_{\mu x. t'}$ contains two identical occurrences of $F_{t'}$) thus avoiding the exponential blow-up required by the intermediate translation into the formula $F_t(q_1, \dots, q_n, y)$. We leave this question open for further research.

We now elaborate the sense in which the function $t(x_1, \dots, x_n)$ is piecewise linear. A *conditioned linear expression* is a pair, written $C \vdash e$, where e is a linear expression, and C is a finite set of strict and non-strict inequalities between linear expressions; i.e., each element of C has one of the forms in (2). We write $C(\vec{r})$ for the conjunction of the inequations obtained by instantiating \vec{r} for \vec{x} in C . The role of a conditioned linear expression, $C \vdash e$, is to specify one piece of a piecewise linear function. The domain of the piece is the set of vectors \vec{r} for which $C(\vec{r})$ is true. And the expression e specifies the linear function that applies over that domain. Note that the domain $\{(r_1, \dots, r_n) \mid C(\vec{r})\}$ is always convex; i.e., if $C(\vec{r})$ and $C(\vec{s})$ then, for all $\lambda \in [0, 1]$, we have $C(\lambda \vec{r} + (1 - \lambda) \vec{s})$. This fact will be exploited in the sequel. We remark that the domain need not be open or closed. It may also have empty interior.

Let \mathcal{F} be a *system* (i.e., finite set) of conditioned linear expressions in variables x_1, \dots, x_n . We say that \mathcal{F} *represents* a function $f: [0, 1]^n \rightarrow [0, 1]$ if the following conditions hold:

1. For all $r_1, \dots, r_n \in [0, 1]$, there exists a conditioned linear expression $(C \vdash e) \in \mathcal{F}$ such that $C(\vec{r})$ is true, and

2. for all $r_1, \dots, r_n \in [0, 1]$, and every conditioned linear expression $(C \vdash e) \in \mathcal{F}$, if $C(\vec{r})$ is true then $e(\vec{r}) = f(\vec{r})$.

Note that, for two conditioned linear expressions $(C_1 \vdash e_1), (C_2 \vdash e_2) \in \mathcal{F}$, we do not require different conditioning sets C_1 and C_2 to be disjoint. However, e_1 and e_2 must agree on any overlap.

Obviously, the function represented by a system of conditioned linear expressions is unique, when it exists. But not every system represents a function. In general, one could impose syntactic conditions on a system to ensure that it represents a function, but we shall not pursue this.

Definition 2.5. We say that a function $f: [0, 1]^n \rightarrow [0, 1]$ is *piecewise linear* if there exists a system \mathcal{F} of conditioned linear expressions in variables x_1, \dots, x_n that represents f . We say that f is *rational piecewise linear* if there exists such an \mathcal{F} containing only rational numbers.

We emphasise that piecewise linear functions, as defined above, need not be continuous (see, e.g., (1) above and Section 3.1 for examples). The result below, which is presumably folklore, justifies the definition we have given.

Proposition 2.6. A function $f: [0, 1]^n \rightarrow [0, 1]$ is piecewise linear if and only if its graph $\{(\vec{x}, y) \in [0, 1]^{n+1} \mid f(\vec{x}) = y\}$ is definable by a formula $F(x_1, \dots, x_n, y)$ in the first-order theory of linear arithmetic. Similarly, it is rational piecewise linear if and only if its graph is definable by a formula of rational linear arithmetic. In the rational case, a defining formula and a representing system of conditioned linear equations can each be computed from the other.

Proof. The proof is a straightforward application of quantifier elimination. Suppose we have a system of k conditioned linear expressions representing f . Each conditioned expression $C \vdash e$ is captured by the implication $(\bigwedge C) \rightarrow y = e$, so the whole system translates into a conjunction of k such implications. To this conjunction, one need only add the range constraints $0 \leq z$ and $z \leq 1$ for each variable z , as further conjuncts. In this way, the graph is easily expressed as a quantifier free formula. (Since the implications are equivalent to disjunctions of atomic formulas, the resulting formula is naturally in conjunctive normal form.)

Conversely, suppose $F(x_1, \dots, x_n, y)$ defines the graph of f . By quantifier elimination, we can assume that F is quantifier free and in disjunctive normal form. Then F is a disjunction of conjunctions, where each conjunction, K , can be easily rewritten in the form

$$\left(\bigwedge C\right) \wedge \left(\bigwedge_{1 \leq i \leq h} y > a_i\right) \wedge \left(\bigwedge_{1 \leq i \leq k} y \geq b_i\right) \wedge \left(\bigwedge_{1 \leq i \leq l} y \leq c_i\right) \wedge \left(\bigwedge_{1 \leq i \leq m} y < d_i\right), \quad (3)$$

such that the only variables in the finite set of atomic formulas C , and linear expressions a_i, b_i, c_i, d_i are x_1, \dots, x_n . Since F is the graph of a function, for all reals r_1, \dots, r_n , there is at most one s such that $K(\vec{r}, s)$ holds, and, if it does, then all of r_1, \dots, r_n, s are in $[0, 1]$. Given such an s , we therefore have:

$$\max\{a_i(\vec{r}) \mid 1 \leq i \leq h\} < \max\{b_i(\vec{r}) \mid 1 \leq i \leq k\} = s = \min\{c_i(\vec{r}) \mid 1 \leq i \leq l\} < \min\{d_i(\vec{r}) \mid 1 \leq i \leq m\}.$$

A system of conditioned linear expressions for f is thus obtained as follows. For each conjunct K in F , written in the form of (3) above, and each j with $1 \leq j \leq k$, include the conditioned linear expression:

$$C, \{b_j > a_i\}_{1 \leq i \leq h}, \{b_j \geq b_i\}_{1 \leq i \leq k}, \{b_j \leq c_i\}_{1 \leq i \leq l}, \{b_j < d_i\}_{1 \leq i \leq m}, \vdash b_j.$$

□

Combining Propositions 2.3 and 2.6 we obtain:

Corollary 2.7. For every Łukasiewicz μ -term $t(x_1, \dots, x_n)$, the function

$$\vec{r} \mapsto t(\vec{r}): [0, 1]^n \rightarrow [0, 1]$$

is piecewise linear. Moreover, if t is a rational μ -term then the function is rational piecewise linear, and a representing system of conditioned linear expressions in variables x_1, \dots, x_n can be computed from t .

McNaughton's Theorem ([24], see also [30]) famously classifies the functions defined by ordinary Łukasiewicz formulas (without scalar multiplication) as the continuous piecewise linear functions on $[0, 1]$ with integer coefficients (McNaughton functions). The result is specialized in [5, Thm. 3.5] where it is shown that monotone McNaughton functions correspond to monotone (i.e., negation free) ordinary Łukasiewicz formulas. Another variant of McNaughton's theorem, proved in [15, Thm. 4.5], classifies rational Łukasiewicz formulas (i.e., with scalar multiplication by rationals) as the continuous piecewise linear functions on $[0, 1]$ with rational coefficients.

A positive answer to the question below would provide an analogous result for the Łukasiewicz μ -calculus.

Question 2.8. *Is every monotone (rational) piecewise linear function $f: [0, 1]^n \rightarrow [0, 1]$ definable by a (rational) μ -term $t(x_1, \dots, x_n)$?*

3 An iterative algorithm for evaluating μ -terms

The computation of a representing system of conditioned linear expressions for a μ -term t via quantifier elimination, provided by the proofs of Propositions 2.3 and 2.6, is indirect. In this section we present an alternative algorithm for calculating the value $t(\vec{r})$ of a rational μ -term at rationals $r_1, \dots, r_n \in [0, 1]$, which is directly based on manipulating conditioned linear expressions. Rather than computing an entire system of conditioned linear expressions representing t , the algorithm works locally to provide a single conditioned expression that applies to the input vector \vec{r} . Fixed points are computed by iterating through approximations to them, starting with 0 for least fixed points and 1 for greatest.

3.1 Motivating the algorithm

The algorithm works inductively on the structure of terms. The crucial aspect is how to compute least and greatest fixed points. Accordingly, given a method for obtaining linear pieces for an inner term $t'(x_1, \dots, x_{n+1})$, we need to specify how to find linear pieces for the fixed-point terms $\mu x_{n+1}. t'$ and $\nu x_{n+1}. t'$.

We illustrate the main idea behind the iteration by considering an example of finding the least-fixed point of the piecewise linear function f specified by:

$$\begin{aligned} x < \frac{1}{2} &\vdash \frac{1}{2}x + \frac{1}{4} \\ \frac{1}{2} \leq x \leq \frac{9}{16} &\vdash x + \frac{1}{8} \\ \frac{9}{16} < x \leq \frac{5}{8} &\vdash \frac{3}{4} \\ \frac{5}{8} \leq x &\vdash \frac{1}{4}x + \frac{19}{32} \end{aligned}$$

This function, which is illustrated in Figure 1, is discontinuous and has a unique fixed-point at $\frac{19}{24}$.

Our algorithm calculates the least fixed-point by iteratively refining a lower-approximation d , starting from $d=0$. We note, however, that a blind iteration of f from 0 does not work. The successive values $0 \leq f(0) \leq f^2(0) \leq f^3(0) \leq \dots$ (i.e., $0 \leq \frac{1}{4} \leq \frac{3}{8} \leq \frac{7}{16} \leq \dots$) never reach a fixed point. Furthermore, their limit $\frac{1}{2}$ is not a fixed point of f , due to the discontinuity of f at $\frac{1}{2}$.

Instead the algorithm works by iterating through the linear pieces. The initial approximation is $d = 0$, and the linear piece for $x = 0$ is retrieved, given by $\frac{1}{2}x + \frac{1}{4}$ on the domain $[0, \frac{1}{2})$. The unique solution of $x = \frac{1}{2}x + \frac{1}{4}$ is $x = \frac{1}{2}$, but this lies outside the domain. So d is replaced by a new approximation, given by $\frac{1}{2}x + \frac{1}{4}$ calculated at the upper bound $x = \frac{1}{2}$ of the domain. That is, the next approximation is $d = \frac{1}{2}$.

We now again retrieve the linear piece at $x = \frac{1}{2}$, which is $x + \frac{1}{8}$ on the domain $[\frac{1}{2}, \frac{9}{16}]$. The equation $x = x + \frac{1}{8}$ has no solution, so d is replaced by the new approximation $x + \frac{1}{8}$ calculated at the upper bound $\frac{9}{16}$, i.e., $d = \frac{11}{16}$.

The linear piece at $x = \frac{11}{16}$ is $\frac{1}{4}x + \frac{19}{32}$ on the domain $[\frac{5}{8}, 1]$ (note that the algorithm has skipped one of the linear pieces of f). The unique solution of $x = \frac{1}{4}x + \frac{19}{32}$ is $x = \frac{19}{24}$. Because this lies in the domain, $\frac{19}{24}$ is the desired least fixed point of f .

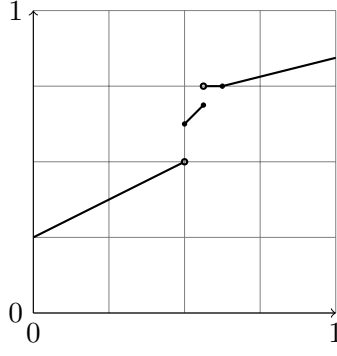


Figure 1: Example discontinuous piecewise linear function

The full algorithm, detailed in Section 3.2 below, adapts the approach outlined above to the general computation of $\mu x_{n+1}.t'$ for $t'(x_1, \dots, x_n, x_{n+1})$. In the case that $n \geq 1$, significant further complications arise due to the need to calculate the fixed point as a linear expression rather than just a number.

3.2 The algorithm

The algorithm takes, as input, a rational μ -term $t(x_1, \dots, x_n)$ and a vector of rationals $(r_1, \dots, r_n) \in [0, 1]^n$, and returns a conditioned linear expression $C \vdash e$, in variables x_1, \dots, x_n , with the following two properties.

(P1) $C(\vec{r})$ is true.

(P2) For all $s_1, \dots, s_n \in \mathbb{R}$, if $C(\vec{s})$ is true then $s_1, \dots, s_n \in [0, 1]$ and $e(\vec{s}) = t(\vec{s})$.

It follows that $e(\vec{r}) = t(\vec{r})$, so e can indeed be used to compute the value $t(\vec{r})$.

For the purposes of the correctness proof in Section 3.5, it is convenient to consider the running of the algorithm in the more general case that r_1, \dots, r_n are arbitrary real numbers in $[0, 1]$. This more general algorithm can be understood as an algorithm in the *Real RAM* model of computation, also known as the *BSS* model after the initials of its proposers [3]. When the input vector is rational, all real numbers encountered during execution of the algorithm are themselves rational, and so the general Real RAM algorithm specialises to a *bona fide* (Turing Machine) algorithm in this case. Moreover, even in the case of irrational inputs, all linear expressions constructed in the course of the algorithm are rational.

The algorithm works recursively on the structure of the term t . We present illustrative cases for terms $t_1 \oplus t_2$ and $\mu x_{n+1}.t'$. The latter is the critical case. The algorithm for $\nu x_{n+1}.t'$ is an obvious dualization.

If t is $t_1 \oplus t_2$ then recursively compute $C_1 \vdash e_1$ and $C_2 \vdash e_2$. If $e_1(\vec{r}) + e_2(\vec{r}) \leq 1$ then return

$$C_1, C_2, e_1 + e_2 \leq 1 \vdash e_1 + e_2 .$$

Otherwise, return

$$C_1, C_2, e_1 + e_2 \geq 1 \vdash 1 .$$

In the case that t is $\mu x_{n+1}.t'$, enter the following loop starting with $D = \emptyset$ and $d = 0$.

Loop: At the entry of the loop we have a finite set D of inequalities between linear expressions in x_1, \dots, x_n , and we have a linear expression $d(x_1, \dots, x_n)$. The loop invariant that applies is:

(I1) $D(\vec{r})$ is true; and

(I2) for all $\vec{s} \in [0, 1]^n$, if $D(\vec{s})$ then $d(\vec{s}) \leq (\mu x_{n+1}.t')(\vec{s})$.

We think of D as constraints propagated from earlier iterations of the loop, and of d as the current approximation to the least fixed point subject to the constraints.

Recursively compute $t'(x_1, \dots, x_{n+1})$ at $(\vec{r}, d(\vec{r}))$ as $C \vdash e$, where e has the form:

$$q_1 x_1 + \dots + q_n x_n + q_{n+1} x_{n+1} + q . \quad (4)$$

In the case that $q_{n+1} \neq 1$, define the linear expression:

$$f := \frac{1}{1 - q_{n+1}} (q_1 x_1 + \dots + q_n x_n + q) . \quad (5)$$

Test if $C(\vec{r}, f(\vec{r}))$ is true. If it is, exit the loop and return:

$$D \cup C(x_1, \dots, x_n, d(x_1, \dots, x_n)) \cup C(x_1, \dots, x_n, f(x_1, \dots, x_n)) \vdash f \quad (6)$$

as the result of the algorithm for $\mu x. t'$ at \vec{r} . Otherwise, if $C(\vec{r}, f(\vec{r}))$ is false, define $N(x_1, \dots, x_n)$ to be the negation of the inequality $e_1(x_1, \dots, x_n, f(x_1, \dots, x_n)) \triangleleft e_2(x_1, \dots, x_n, f(x_1, \dots, x_n))$ (using \triangleleft to stand for either $<$ or \leq), where $e_1(x_1, \dots, x_{n+1}) \triangleleft e_2(x_1, \dots, x_{n+1})$ is a chosen inequality in C for which $e_1(\vec{r}, f(\vec{r})) \triangleleft e_2(\vec{r}, f(\vec{r}))$ is false, and go to **find next approximation** below.

In the case that $q_{n+1} = 1$, test the equality $q_1 r_1 + \dots + q_n r_n + q = 0$. If true, exit the loop with result:

$$D \cup C(x_1, \dots, x_n, d(x_1, \dots, x_n)) \cup \{q_1 x_1 + \dots + q_n x_n + q = 0\} \vdash d . \quad (7)$$

If instead $q_1 r_1 + \dots + q_n r_n + q \neq 0$, choose $N(x_1, \dots, x_n)$ to be whichever of the inequalities

$$q_1 x_1 + \dots + q_n x_n + q < 0 \quad 0 < q_1 x_1 + \dots + q_n x_n + q$$

is true for \vec{r} , and proceed with **find next approximation** below.

Find next approximation: Arrange the inequalities in C so they have the following structure.

$$C' \cup \{x_{n+1} > a_i\}_{1 \leq i \leq l'} \cup \{x_{n+1} \geq a_i\}_{l' < i \leq l} \cup \{x_{n+1} \leq b_i\}_{1 \leq i \leq m'} \cup \{x_{n+1} < b_i\}_{m' < i \leq m} \quad (8)$$

such that the only variables in the inequalities C' , and linear expressions a_i, b_i are x_1, \dots, x_n . Choose j with $1 \leq j \leq m$ such that $b_j(\vec{r}) \leq b_i(\vec{r})$ for all i with $1 \leq i \leq m$ (in the sequel we shall refer to b_j as the *infimum term*). Then go back to **loop**, taking

$$D \cup C(x_1, \dots, x_n, d(x_1, \dots, x_n)) \cup \{N(x_1, \dots, x_n)\} \cup \{b_j \leq b_i \mid 1 \leq i \leq m\} \quad e(\vec{x}, b_j(\vec{x})) \quad (9)$$

to replace D and d respectively.

3.3 A simple example

Consider the rational μ -term $t = \mu x. (\mathbb{P}_{\geq \frac{1}{2}} x \sqcup \frac{1}{2})$, where $\mathbb{P}_{\geq \frac{1}{2}} x$ is the macro formula as in Definition 2.1, that is $\mathbb{P}_{\geq \frac{1}{2}} x = \mathbb{P}_{=1}(x \oplus \frac{1}{2}) = \nu y. (y \odot (x \oplus \frac{1}{2}))$. Thus,

$$t = \mu x. \left(\nu y. (y \odot (x \oplus \frac{1}{2})) \sqcup \frac{1}{2} \right)$$

Here, $t'(x) = \nu y. (y \odot (x \oplus \frac{1}{2})) \sqcup \frac{1}{2}$ is a discontinuous function, and the value of t is 1.

We omit giving a detailed simulation of the algorithm on the subexpression $t'(x)$ at $x = r$. The result it produces, however, is $\{0 \leq x < \frac{1}{2}\} \vdash \frac{1}{2}$ if $r < \frac{1}{2}$, and $\{\frac{1}{2} \leq x \leq 1\} \vdash 1$ if $r \geq \frac{1}{2}$.

We run the algorithm on input $\mu x. t'(x)$. Set $D = \emptyset$ and $d = 0$. Calculating $t'(x)$ at $x = 0$ we obtain $C \vdash e$ as $\{0 \leq x < \frac{1}{2}\} \vdash \frac{1}{2}$. The value of the coefficient of x in e is 0. Thus, we need to calculate $f := \frac{1}{1-0}(\frac{1}{2}) = \frac{1}{2}$. The constraint $C(\frac{1}{2})$ does not hold. Thus we need to iterate the algorithm with $D = \emptyset$ and $d = \frac{1}{2}$. Calculating $t'(x)$ at $x = \frac{1}{2}$ produces $C \vdash e$ as $\{\frac{1}{2} \leq x \leq 1\} \vdash 1$. Compute $f := \frac{1}{1-0}(1) = 1$. Since $C(1)$ holds, the algorithm terminates with $\emptyset \vdash 1$, as desired.

3.4 A more complex example

As a second example, we compute the value of the μ -term $\nu x_0. \mu x_1. (\frac{5}{8} \oplus \frac{3}{8} x_0) \odot (\frac{1}{2} \sqcup (\frac{3}{8} \oplus \frac{1}{2} x_1))$ to describe in some detail the behavior of the algorithm on nested fixed-point expressions.

We illustrate the execution using subscripts such as d_0, d_1 as reminders of the current depth in the nested recursions that the algorithm is at. So, e.g., d_0 is the current approximation within the loop evaluating the $\nu x_0.(\dots)$ expression, and d_1 is the approximation in the inner loop evaluating the $\mu x_1.(\dots)$ subexpression.

The algorithm starts computing the value of the outer greatest fixed-point expression starting from $D_0 = \emptyset$ and approximating function $d_0 := 1$. The conditional linear expression $C_0 \vdash e_0$ associated with the inner μ -term $\mu x_1. (\frac{5}{8} \oplus \frac{3}{8} x_0) \odot (\frac{1}{2} \sqcup (\frac{3}{8} \oplus \frac{1}{2} x_1))$ at value $(x_0 = 1)$ needs to be calculated.

This is computed iteratively starting from the condition $D_1 = \emptyset$ and $d_1 = 0$. Recursively computing $(\frac{5}{8} \oplus \frac{3}{8} x_0) \odot (\frac{1}{2} \sqcup (\frac{3}{8} \oplus \frac{1}{2} x_1))$ at values $(x_0 = 1, x_1 = 0)$ one obtains (details are omitted) the conditional linear expression $C_1 \vdash e_1$ with $C_1 = \{0 \leq x_0 \leq 1, x_1 \leq \frac{1}{4}\}$ and $e_1 = \frac{3}{8} x_0 + \frac{1}{8}$. Note that the coefficient q_1 of the variable x_1 , see (4), is 0. The function f_1 , see (5), is then defined as $f_1(x_0) = \frac{3}{8} x_0 + \frac{1}{8}$. The condition $C_1(1, f_1(1)) = C_1(1, \frac{1}{2})$ does not hold, since $\frac{1}{2} \not\leq \frac{1}{4}$. Thus, the inequality $N_1(x_0) := \frac{3}{8} x_0 + \frac{1}{8} > \frac{1}{4}$, which is the negation of $f_1(x_0) \leq \frac{1}{4}$, is calculated. Simplifying, we have $N_1(x_0) := x_0 > \frac{1}{3}$. The supremum term in C_1 , dual to the infimum term in (8), is $b_j(x_0) := \frac{1}{4}$, using the notation of (8). So the loop is repeated, see (9), with $D_1 := \{0 \leq x_0 \leq 1, d_1 \leq \frac{1}{4}, x_0 > \frac{1}{3}\} = \{\frac{1}{3} < x_0 < \frac{1}{4}\}$ and $d_1 := \frac{3}{8} x_0 + \frac{1}{8}$.

Computing $(\frac{5}{8} \oplus \frac{3}{8} x_0) \odot (\frac{1}{2} \sqcup (\frac{3}{8} \oplus \frac{1}{2} x_1))$ with values $(x_0 = 1, x_1 = d_1(1) = \frac{1}{2})$ one obtains the conditional linear expression $C_1 \vdash e_1$ with $C_1 = \{0 \leq x_0 \leq 1, \frac{1}{4} \leq x_1 \leq 1\}$ and $e_1 = \frac{3}{8} x_0 + \frac{1}{2} x_1$. As in the previous iteration, $q_1 = \frac{1}{2} \neq 0$, and the function $f_1(x_0)$ is then defined as $f_1(x_0) = \frac{3}{4} x_0$. The algorithm checks that $C_1(1, f_1(1)) = C_1(1, \frac{3}{4})$ holds and terminates with result $C_0 \vdash e_0$ equals to $\frac{1}{3} < x_0 \leq 1, \frac{1}{4} \leq \frac{3}{8} x_0 + \frac{1}{8} \leq 1, \frac{1}{4} \leq \frac{3}{4} x_0 \leq 1 \vdash \frac{3}{4} x_0$.

Simplifying, $C_0 \vdash e_0$ equals $\frac{1}{3} < x_0 \leq 1 \vdash \frac{3}{4} x_0$. Since q_0 (the coefficient of the x_0 variable) is $\frac{3}{4}$, the function $f_0 := 0$ is defined. The condition $C_0(f_0()) = C_0(0)$ is not satisfied. Calculating the negated condition $N_0()$ is redundant, since the index is 0. The infimum term in C_0 is $b_j() := \frac{1}{3}$, again using the notation of (8). Therefore the loop is repeated with $D_0 = \emptyset$ and $d_0 := \frac{3}{4} \cdot \frac{1}{3}$.

The conditional linear expression associated with the μ -term $\mu x_1. (\frac{5}{8} \oplus \frac{3}{8} x_0) \odot (\frac{1}{2} \sqcup (\frac{3}{8} \oplus \frac{1}{2} x_1))$ at value $(x_0 = \frac{1}{4})$ needs to be calculated. This time we omit the details of the computation which produces the result: $C_0 \vdash e_0$ with $C_0 = \{0 \leq x_0 \leq \frac{1}{3}\}$ and $e_0(x_0) = \frac{3}{8} x_0 + \frac{1}{8}$.

The coefficient $q_0 = \frac{3}{8}$ is not equal to 1, thus the function $f_0() := \frac{1}{5}$ is defined. The condition $C_0(f_0()) = C_0(\frac{1}{5})$ holds. Thus the algorithm returns the conditional linear expression $C_0(f_0) \cup C_0(d_0) \vdash \frac{1}{5}$, i.e., $\emptyset \vdash \frac{1}{5}$ as result. That is, the result is $\frac{1}{5}$.

3.5 Correctness of the algorithm

Theorem 3.1. *Let $t(x_1, \dots, x_n)$ be any Lukasiewicz μ -term. Then, for every input vector $(r_1, \dots, r_n) \in [0, 1]^n$, the above (Real RAM) algorithm terminates with a conditioned linear expression $C_{\vec{r}} \vdash e_{\vec{r}}$ satisfying properties (P1) and (P2). Moreover, the set of all possible resulting conditioned linear expressions*

$$\{C_{\vec{r}} \vdash e_{\vec{r}} \mid \vec{r} \in [0, 1]^n\} \quad (10)$$

is finite, and thus provides a representing system for the function $t: [0, 1]^n \rightarrow [0, 1]$.

Before the proof it is convenient to introduce some terminology associated with the properties stated in the theorem. For a μ -term t , we call the cardinality of the set (10) of possible results, $C_{\vec{r}} \vdash e_{\vec{r}}$, the *basis size*, and we call the maximum number of inequalities in any $C_{\vec{r}}$ the *condition size*.

Proof. The proof is by induction on the structure of t . We verify the critical case in which t is $\mu x_{n+1}. t'$.

We show first that the loop invariants (I1), (I2) guarantee that any result returned via (6) or (7) satisfies (P1) and (P2). By induction hypothesis, the recursive computation of $t'(x_1, \dots, x_{n+1})$ at $(\vec{r}, d(\vec{r}))$ as $C \vdash e$, where e has the form $q_1 x_1 + \dots + q_n x_n + q_{n+1} x_{n+1} + q$ as in (4), satisfies: $C(\vec{r}, d(\vec{r}))$; and, for all $s_1, \dots, s_{n+1} \in \mathbb{R}$, if $C(s_1, \dots, s_{n+1})$ then $\vec{s} \in [0, 1]^n$ and $t'(s_1, \dots, s_{n+1}) = e(s_1, \dots, s_{n+1})$.

In the case that $q_{n+1} \neq 1$, the linear expression f , defined in (5), maps any $s_1, \dots, s_n \in \mathbb{R}$ to the unique solution $f(\vec{s})$ to the equation $x_{n+1} = e(s_1, \dots, s_n, x_{n+1})$ in \mathbb{R} . Suppose that $D(\vec{s})$ holds. Then, by loop invariant (I2), $d(\vec{s}) \leq (\mu x_{n+1}. t')(\vec{s})$. Suppose also that $C(\vec{s}, f(\vec{s}))$. Then $t'(\vec{s}, f(\vec{s})) =$

$e(\vec{s}, f(\vec{s})) = f(\vec{s})$, i.e., $f(\vec{s})$ is a fixed point of $x_{n+1} \mapsto t'(\vec{s}, x_{n+1})$; whence, $(\mu x_{n+1}. t')(\vec{s}) \leq f(\vec{s})$. Suppose, finally, that $C(\vec{s}, d(\vec{s}))$ also holds. Then, because both $C(\vec{s}, d(\vec{s}))$ and $C(\vec{s}, f(\vec{s}))$, and $d(\vec{s}) \leq (\mu x_{n+1}. t')(\vec{s}) \leq f(\vec{s})$, we have, by the convexity of constraints, that $t'(\vec{s}, s_{n+1}) = e(\vec{s}, s_{n+1})$ for all $s_{n+1} \in [d(\vec{s}), f(\vec{s})]$. So $f(\vec{s})$ is the unique fixed-point of $x_{n+1} \mapsto t'(\vec{s}, x_{n+1})$ on $[d(\vec{s}), f(\vec{s})]$. Since, $d(\vec{s}) \leq (\mu x_{n+1}. t')(\vec{s})$, we have $f(\vec{s}) = (\mu x_{n+1}. t')(\vec{s})$. This argument justifies that the conditioned linear expression of (6) satisfies (P2). It satisfies (P1) just if $C(\vec{r}, f(\vec{r}))$, which is exactly the condition under which (6) is returned as the result.

In the case that $q_{n+1} = 1$ then, for any $s_1, \dots, s_n \in \mathbb{R}$, the equation $x_{n+1} = e(s_1, \dots, s_n, x_{n+1})$ has a solution if and only if $q_1 s_1 + \dots + q_n s_n + q = 0$, in which case any $x_{n+1} \in \mathbb{R}$ is a solution. Suppose that $q_1 s_1 + \dots + q_n s_n + q = 0$ and $C(\vec{s}, d(\vec{s}))$ both hold. Then $t'(s_1, \dots, s_n, d(\vec{s})) = e(\vec{s}, d(\vec{s})) = d(\vec{s})$, so $d(\vec{x})$ is a fixed point of $x_{n+1} \mapsto t'(\vec{s}, x_{n+1})$. If also $D(\vec{s})$ holds then, by loop invariant (I2), $d(\vec{x}) = (\mu x_{n+1}. t')(\vec{s})$. We have justified that the conditioned linear expression of (7) satisfies (P2). It satisfies (P1) just if $q_1 r_1 + \dots + q_n r_n + q = 0$, which is exactly the condition under which (7) is returned as the result.

Next we show that the loop invariants are preserved through the computation. Properties (I1) and (I2) are trivially satisfied by the initial values $D = \emptyset$ and $d = 0$. We must show that they are preserved when D and d are modified via (9), which happens when execution passes to **find next approximation**. In this subroutine, the inequalities in C are first arranged as in (8) where, as $C(\vec{r}, d(\vec{r}))$, we must have $m \geq 1$, as otherwise $C(\vec{r}, s)$ would hold for all real $s \geq d(\vec{r})$, contradicting that $C(\vec{r}, s)$ implies $s \in [0, 1]$. (Similarly, $l \geq 1$.) Thus there indeed exists j with $1 \leq j \leq m$ such that $b_j(\vec{r}) \leq b_i(\vec{r})$ for all i with $1 \leq i \leq m$. It is immediate that the constraints in the modified D of (9) are true for \vec{r} . Thus (I1) is preserved. To show (I2), suppose s_1, \dots, s_n satisfy the constraints, i.e.,

$$D(\vec{s}) \quad C(\vec{s}, d(\vec{s})) \quad N(\vec{s}) \quad \{b_j(\vec{s}) \leq b_i(\vec{s}) \mid 1 \leq i \leq m\}.$$

Defining $r' = (\mu x_{n+1}. t')(\vec{s})$, by (I2) for D, d we have $d(\vec{s}) \leq r'$. We must show that $e(\vec{s}, b_j(\vec{s})) \leq r'$. By the definition of $N(x_1, \dots, x_n)$, in either the $q_{n+1} \neq 1$ or $q_{n+1} = 1$ case, $N(\vec{s})$ implies that $C(\vec{s}, r')$ does not hold. Because $C(\vec{s}, d(\vec{s}))$ and by the choice of j , it holds that $C(\vec{s}, s)$, for all $s \in [0, 1]$ such that $s = d(\vec{s})$ or $d(\vec{s}) < s < b_j(\vec{s})$. Since $C(\vec{s}, r')$ is false and $d(\vec{s}) \leq r'$, it follows from the convexity of the conditioning set C that, for every s with $s = d(\vec{s})$ or $d(\vec{s}) < s < b_j(\vec{s})$, we have $s < r'$. Whence, since r' is the least prefixed point for $x_{n+1} \mapsto t'(\vec{s}, x_{n+1})$, also $s < t'(\vec{s}, s) \leq r'$, i.e.,

$$s < e(\vec{s}, s) \leq r'. \quad (11)$$

Thus, $e(\vec{s}, b_j(\vec{s})) = \sup\{e(\vec{s}, s) \mid s = d(\vec{s}) \text{ or } d(\vec{s}) \leq s < b_j(\vec{s})\} \leq r'$. Thus, $e(\vec{s}, b_j(\vec{s})) \leq r'$, i.e., it is an approximation to the fixed point. Moreover, it is a good new approximation to choose in the sense that:

$$d(\vec{s}) < e(\vec{s}, b_j(\vec{s})) \text{ and } \text{not } C(\vec{s}, e(\vec{s}, b_j(\vec{s}))). \quad (12)$$

The former holds because $d(\vec{s}) < e(\vec{s}, d(\vec{s}))$, by (11), and $d(\vec{s}) \leq b_j(\vec{s})$. The latter because if $C(\vec{s}, e(\vec{s}, b_j(\vec{s})))$ then, in particular, $e(\vec{s}, b_j(\vec{s})) \leq b_j(\vec{s})$, so $b_j(\vec{s}) = e(\vec{s}, b_j(\vec{s})) = r'$, contradicting that not $C(\vec{s}, r')$.

To show termination, by induction hypothesis, collecting all possible results of running the algorithm on t' produces a representing system for t' : $[0, 1]^{n+1} \rightarrow [0, 1]$:

$$C_1 \vdash e_1 \quad \dots \quad C_{k'} \vdash e_{k'} \quad , \quad (13)$$

where k' is the basis size of t' . We now analyse the execution of the algorithm for $\mu x_{n+1}. t'$ on a given input vector (r_1, \dots, r_n) . On iteration number i , the loop is entered with constraints D_i and approximation d_i (where $D_1 = \emptyset$ and $d_1 = 0$), after which the recursive call to the algorithm for t' yields one of the conditioned linear expressions, $C_{k_i} \vdash e_{k_i}$, from (13) above, such that $C_{k_i}(\vec{r}, d_i(\vec{r}))$ holds. Then, depending on conditions involving only $C_{k_i} \vdash e_{k_i}$ and \vec{r} , either a result is returned, or D_{i+1} and d_{i+1} are constructed for the loop to be repeated. By (12), at iteration $i + 1$ of the loop, we have $d_{i+1}(\vec{r}) > d_i(\vec{r})$ and also $C_{k_i}(\vec{r}, d_{i+1}(\vec{r}))$ is false. Since each conditioning set is convex, it follows that no C_j can occur twice in the list C_{k_1}, C_{k_2}, \dots . Hence the algorithm must exit the loop after at most k' iterations. Therefore, the computation for $\mu x. t'$ at \vec{r} terminates.

It remains to show that the algorithm for $\mu x.t'$ produces only finitely many conditioned linear expressions $C_{\vec{r}} \vdash e_{\vec{r}}$.

We analyse the control flow in the algorithm for $\mu x_{n+1}.t'$ on a given input vector (r_1, \dots, r_n) . On iteration number i , the loop is entered with constraints D_i and approximation d_i , after which the recursive call to the algorithm for t' yields one of the conditioned linear expressions, $C_{k_i} \vdash e_{k_i}$. Suppose that C_{k_i} and D_i contain u and v inequalities respectively. If the loop is exited producing (6) as result then the resulting $C_{\vec{r}}$ has $2u + v$ inequalities. If it is exited producing (7) as result then $C_{\vec{r}}$ has $u + v + 2$ inequalities (where $u + v + 2 \leq 2u + v$ because C_{k_i} has to enforce the range constraint $0 \leq x_{n+1} \leq 1$). Otherwise, the algorithm repeats the loop, entering iteration $i + 1$ with D_{i+1} , given by (9), having at most $2u + v$ inequalities (N contributes 1 inequality, and there are at most $u - 1$ inequalities $b_j \leq b_i$ in (9) since $l \geq 1$).

Therefore, if l' is now the maximum number of inequalities occurring in any C_j from (13) (i.e., if it is the condition size for t') the algorithm for $\mu x_{n+1}.t'$ at \vec{r} , which runs for at most k' iterations, results in $C_{\vec{r}}$ containing at most $2k'l'$ inequalities.

To bound the number of results $C_{\vec{r}} \vdash e_{\vec{r}}$, we count the possible control flows of the algorithm. At iteration i , the algorithm uses $C_{k_i} \vdash e_{k_i}$ from (13), using which it might terminate with either (6) or (7), or it might repeat the loop, entering iteration $i + 1$ with D_{i+1} , given by (9), which can arise from C_{k_i} in a number of ways determined by the possible pairs of choices for N and b_j in (9). In the case that the variable vector (x_1, \dots, x_n) is empty (i.e., the term $\mu x_{n+1}.t'$ is closed) the constraints in D are redundant (they are simply true inequalities between rationals) and so can be discarded. In the case that $n \geq 1$, there are at least 2 inequalities in C giving range constraints on x_1 , so there are at most l' choices for N ($l' - 2$ choices in the case that $q_{n+1} \neq 1$, and 2 in the case $q_{n+1} = 1$). Irrespective of n , there are at most $l' - 1$ choices for b_j (taking n into account this can be improved to $l' - 2n - 1$). Therefore, the execution of the algorithm, is determined by the sequence:

$$k_1, u_1, k_2, u_2, \dots, k_m, v$$

where: $m \leq k'$ is the number of loop iterations performed; each u_i , where $1 \leq u_i \leq l'(l' - 1)$, represents the choice of N and b_j used in the construction of D_{i+1} (9), and v is 1 or 2 according to whether the resulting $C_{\vec{r}} \vdash e_{\vec{r}}$ is returned via (6) or (7). Since each number k_i is distinct, the number of different such sequences is bounded by:

$$2 \sum_{m=1}^{k'} \frac{k'!}{(k' - m)!} (l'(l' - 1))^{m-1} \leq (k'(l')^2)^{k'} \quad , \quad (14)$$

where the right-hand-side gives a somewhat loose upper bound. Therefore, the number of possible results $C_{\vec{r}} \vdash e_{\vec{r}}$ for the algorithm for $\mu x_{n+1}.t'$ is at most $(k'(l')^2)^{k'}$. \square

The above proof gives a truly abysmal complexity bound for the algorithm. Let the basis and condition size for the term $t'(x_1, \dots, x_{n+1})$ be k' and l' respectively. Then, as in the proof, the basis and condition size for $\mu x_{n+1}.t'$ are respectively bounded by:

$$k \leq (k'(l')^2)^{k'} \quad \text{and} \quad l \leq 2k'l' \quad .$$

Using these bounds, the basis and condition size have non-elementary growth in the number of fixed points in a term t .

3.6 Comparison

According to the crude complexity analyses we have given, the evaluation of Łukasiewicz μ -terms via rational linear arithmetic and (naive) black-box quantifier elimination is (in having doubly- and triply-exponential space and time complexity bounds) preferable to the (non-elementary space and hence time) evaluation via the direct algorithm. Nevertheless, we expect the direct algorithm to work better than this in practice. Indeed, a main motivating factor in the design of the direct algorithm is that the algorithm for $\mu x_{n+1}.t'$ only explores as much of the basis set for t' as it needs to, and does so in an order that is

tightly constrained by the monotone improvements made to the approximating d expressions along the way. In contrast, the crude complexity analysis is based on a worst-case scenario in which the algorithm is assumed to visit the entire basis for t' , and, moreover, to do so, for different input vectors \vec{r} , in every possible order for visiting the different basis sets. Perhaps better bounds can be obtained by a combination of relatively straightforward optimisations of the algorithm and more careful analysis.

4 Łukasiewicz Modal μ -Calculus

In this section we introduce a modal extension of the Łukasiewicz μ -Calculus whose formulas are interpreted over *probabilistic nondeterministic transition systems* also known as *Markov decision processes* (see, e.g., Section 10.6 of [1]).

Definition 4.1. Given a set S we denote with $\mathcal{D}(S)$ the set of (*discrete*) *probability distributions* on S defined as $\mathcal{D}(S) = \{d : S \rightarrow [0, 1] \mid \sum_{s \in S} d(s) = 1\}$. We say that $d \in \mathcal{D}(S)$ is *rational* if $d(s)$ is a rational number, for all $s \in S$.

Definition 4.2. A *probabilistic nondeterministic transition system* (PNTS) is a pair (S, \rightarrow) where S is a set of states and $\rightarrow \subseteq S \times \mathcal{D}(S)$ is the *accessibility* relation. We write $s \not\rightarrow$ if $\{d \mid s \rightarrow d\} = \emptyset$. A PNTS (S, \rightarrow) is *finite rational* if S is finite and $\bigcup_{s \in S} \{d \mid s \rightarrow d\}$ is a finite set of rational probability distributions.

We now introduce the Łukasiewicz Modal μ -Calculus which extends the probabilistic (or quantitative) modal μ -calculus (qL μ) of [17, 29, 23, 8].

Definition 4.3. The syntax of formulas of Łukasiewicz Modal μ -Calculus is generated by the following grammar:

$$\phi ::= X \mid P \mid \overline{P} \mid q\phi \mid \phi \sqcup \psi \mid \phi \sqcap \psi \mid \phi \oplus \psi \mid \phi \odot \psi \mid \Diamond\phi \mid \Box\phi \mid \mu X. \phi \mid \nu X. \phi ,$$

where q ranges over rationals in $[0, 1]$, X over a countable set **Var** of variables and P over a set **Prop** of propositional letters which come paired with associated complements \overline{P} . As a convention we denote with $\underline{1}$ the formula $\nu X. X$ and with \underline{q} the formula $q \underline{1}$.

Thus, beside the addition of the modal operators \Diamond and \Box , we also added atomic propositional letters and, to adhere with well-established notation, we used upper-case letters for the variables and greek letters for formulas. Without introducing any significant ambiguity, we also write L μ as a shorthand for Łukasiewicz Modal μ -Calculus. For mild convenience in the encoding of **PCTL** below, we consider a version with unlabelled modalities and propositional letters. However, the approach of this paper easily adapts to a labeled version of L μ .

The probabilistic (or quantitative) modal μ -calculus (qL μ) of [17, 29, 23, 8] is the fragment of L μ obtained by removing the Łukasiewicz connectives (\odot, \oplus) and *scalar multiplications* ($q\phi$) by rationals numbers in $[0, 1]$.

Łukasiewicz Modal μ -Calculus formulas are interpreted over PNTS's as we now describe.

Definition 4.4. Given a PNTS (S, \rightarrow) , an *interpretation* for the variables and propositional letters is a function $\rho : (\mathbf{Var} \uplus \mathbf{Prop}) \rightarrow (S \rightarrow [0, 1])$ such that $\rho(\overline{P})(x) = 1 - \rho(P)(x)$. Given a function $f : S \rightarrow [0, 1]$ and $X \in \mathbf{Var}$ we define the interpretation $\rho[f/X]$ as $\rho[f/X](X) = f$ and $\rho[f/X](Y) = \rho(Y)$, for $X \neq Y$.

Definition 4.5. The semantics of a L μ formula ϕ interpreted over (S, \rightarrow) with interpretation ρ is a function $\llbracket \phi \rrbracket_\rho : S \rightarrow [0, 1]$ defined inductively on the structure of ϕ as follows:

$$\begin{array}{ll} \llbracket X \rrbracket_\rho = \rho(X) & \llbracket q\phi \rrbracket_\rho(x) = q \cdot \llbracket \phi \rrbracket_\rho(x) \\ \llbracket P \rrbracket_\rho = \rho(P) & \llbracket \overline{P} \rrbracket_\rho = 1 - \rho(P) \\ \llbracket \phi \sqcup \psi \rrbracket_\rho(x) = \max\{\llbracket \phi \rrbracket_\rho(x), \llbracket \psi \rrbracket_\rho(x)\} & \llbracket \phi \sqcap \psi \rrbracket_\rho(x) = \min\{\llbracket \phi \rrbracket_\rho(x), \llbracket \psi \rrbracket_\rho(x)\} \\ \llbracket \phi \oplus \psi \rrbracket_\rho(x) = \min\{1, \llbracket \phi \rrbracket_\rho(x) + \llbracket \psi \rrbracket_\rho(x)\} & \llbracket \phi \odot \psi \rrbracket_\rho(x) = \max\{0, \llbracket \phi \rrbracket_\rho(x) + \llbracket \psi \rrbracket_\rho(x) - 1\} \\ \llbracket \Diamond\phi \rrbracket_\rho(x) = \bigsqcup_{x \rightarrow d} \left(\sum_{y \in S} d(y) \llbracket \phi \rrbracket_\rho(y) \right) & \llbracket \Box\phi \rrbracket_\rho(x) = \prod_{x \rightarrow d} \left(\sum_{y \in S} d(y) \llbracket \phi \rrbracket_\rho(y) \right) \\ \llbracket \mu X. \phi \rrbracket_\rho = \text{lfp} (f \mapsto \llbracket \phi \rrbracket_{\rho[f/X]}) & \llbracket \nu X. \phi \rrbracket_\rho = \text{gfp} (f \mapsto \llbracket \phi \rrbracket_{\rho[f/X]}) \end{array}$$

As in Section 2, the interpretation of every operator is monotone, thus the existence of least and greatest points in the last two clauses is guaranteed by the Knaster-Tarski theorem.

The operation $\text{negate}(_)$ (see Section 2) is extended to $L\mu$ modal formulas by defining $\text{negate}(\Diamond\phi) = \Box(\text{negate}(\phi))$ and $\text{negate}(\Box\phi) = \Diamond(\text{negate}(\phi))$ as expected.

5 Encoding PCTL in Łukasiewicz Modal μ -Calculus

In this section, we show that the Łukasiewicz Modal μ -Calculus of the previous section can encode the logic **PCTL**. We refer the reader to Section 10.6.2 of [1] for an extensive presentation of PCTL.

5.1 Summary of PCTL

The notions of *paths*, *schedulers* and *Markov runs* in a PNTS are at the basis of the logic **PCTL**.

Definition 5.1. For a given PNTS $\mathcal{L} = (S, \rightarrow)$ the binary relation $\rightsquigarrow_{\mathcal{L}} \subseteq S \times S$ is defined as follows: $\rightsquigarrow_{\mathcal{L}} = \{(s, t) \mid \exists d.(s \rightarrow d \wedge d(t) > 0)\}$. Note that $s \not\rightsquigarrow$ if and only if $s \not\rightsquigarrow$. We refer to (S, \rightsquigarrow) as the *graph underlying* \mathcal{L} .

Definition 5.2. A *path* in a PNTS $\mathcal{L} = (S, \rightarrow)$ is an ordinary path in the graph (S, \rightsquigarrow) , i.e., a finite or infinite sequence $\{s_i\}_{i \in I}$ of states such that $s_i \rightsquigarrow s_{i+1}$, for all $i + 1 \in I$. We say that a path is *maximal* if either it is infinite or it is finite and its last entry is a state s_n without successors, i.e., such that $s_n \not\rightsquigarrow$. We denote with $P(\mathcal{L})$ the set of all maximal paths in \mathcal{L} . The set $P(\mathcal{L})$ is endowed with the topology generated by the basic open sets $U_{\vec{s}} = \{\vec{r} \mid \vec{s} \sqsubseteq \vec{r}\}$ where \vec{s} is a finite sequence of states and \sqsubseteq denotes the prefix relation on sequences. The space $P(\mathcal{L})$ is always 0-dimensional, i.e., the basic sets $U_{\vec{s}}$ are both open and closed and thus form a Boolean algebra. We denote with $P(s)$ the open set $U_{\{s\}}$ of all maximal paths having s as first state.

Definition 5.3. A *scheduler* in a PNTS (S, \rightarrow) is a partial function σ from non-empty finite sequences $s_0 \dots s_n$ of states to probability distributions $d \in \mathcal{D}(S)$ such that $\sigma(s_0 \dots s_n)$ is not defined if and only if $s_n \not\rightsquigarrow$ and, if σ is defined at $s_0 \dots s_n$ with $\sigma(s_0 \dots s_n) = d$, then $s_n \rightarrow d$ holds. A pair (s, σ) is called a *Markov run* in \mathcal{L} and denoted by M_{σ}^s . It is clear that each Markov run M_{σ}^s can be identified with a (generally) infinite Markov chain (having a tree structure) whose vertices are finite sequences of states and having $\{s\}$ as root.

Markov runs are useful as they naturally induce probability measures on the space $P(\mathcal{L})$.

Definition 5.4. Let $\mathcal{L} = (S, \rightarrow)$ be a PNTS and M_{σ}^s a Markov run. We define the measure m_{σ}^s on $P(\mathcal{L})$ as the unique (by Carathéodory extension theorem) measure specified by the following assignment of basic open sets:

$$m_{\sigma}^s(U_{s_0 \dots s_n}) = \prod_{i=0}^{n-1} d_i(s_{i+1})$$

where $d_i = \sigma(s_0 \dots s_i)$ and $\prod \emptyset = 1$. It is simple to verify that m_{σ}^s is a probability measure, i.e., $m_{\sigma}^s(P(\mathcal{L})) = 1$. We refer to m_{σ}^s as the probability measure on $P(\mathcal{L})$ induced by the Markov run M_{σ}^s .

We are now ready to specify the syntax and semantics of **PCTL**.

Definition 5.5. Let the letter P range over a countable set of propositional symbols **Prop**. The class of **PCTL state-formulas** ϕ is generated by the following two-sorted grammar:

$$\phi ::= \text{true} \mid P \mid \neg\phi \mid \phi \vee \phi \mid \exists\psi \mid \forall\psi \mid \mathbb{P}_{\bowtie q}^{\exists}\psi \mid \mathbb{P}_{\bowtie q}^{\forall}\psi$$

with $q \in \mathbb{Q} \cap [0, 1]$ and $\bowtie \in \{>, \geq\}$, where *path-formulas* ψ are generated by the simple grammar: $\psi ::= \circ\phi \mid \phi_1 \mathcal{U} \phi_2$. Adopting standard terminology, we refer to the connectives \circ and \mathcal{U} as the *next* and *until* operators, respectively.

Definition 5.6. Given a PNTS (S, \rightarrow) , a **PCTL-interpretation** for the propositional letters is a function $\rho : \text{Prop} \rightarrow 2^S$, where 2^S denotes the powerset of S .

Definition 5.7. Given a PNTS (S, \rightarrow) and a **PCTL**-interpretation ρ for the propositional letters, the semantics $\llbracket \phi \rrbracket_\rho$ of a **PCTL** state-formula ϕ is a subset of S

- $\llbracket \text{true} \rrbracket_\rho = S$, $\llbracket P \rrbracket_\rho = \rho(P)$, $\llbracket \phi_1 \vee \phi_2 \rrbracket_\rho = \llbracket \phi_1 \rrbracket_\rho \cup \llbracket \phi_2 \rrbracket_\rho$, $\llbracket \neg \phi \rrbracket_\rho = S \setminus \llbracket \phi \rrbracket_\rho$,
- $s \in \llbracket \exists \psi \rrbracket_\rho$ if and only there exists $\vec{s} \in P(s)$ such that $\vec{s} \in \llbracket \psi \rrbracket_\rho$
- $s \in \llbracket \forall \psi \rrbracket_\rho$ if and only for all $\vec{s} \in P(s)$ it holds that $\vec{s} \in \llbracket \psi \rrbracket_\rho$
- $s \in \llbracket \mathbb{P}_{\bowtie q}^\exists \psi \rrbracket_\rho$ if and only $(\bigsqcup_\sigma m_\sigma^s(\llbracket \psi \rrbracket_\rho)) \bowtie q$
- $s \in \llbracket \mathbb{P}_{\bowtie q}^\forall \psi \rrbracket_\rho$ if and only $(\bigsqcap_\sigma m_\sigma^s(\llbracket \psi \rrbracket_\rho)) \bowtie q$

where σ ranges over schedulers and the semantics $\llbracket \psi \rrbracket_\rho$ of path formulas is a subset of $P(\mathcal{L})$ defined as:

- $\vec{s} \in \llbracket \circ \phi \rrbracket_\rho$ if and only if $|\vec{s}| \geq 2$ (i.e., $\vec{s} = s_0.s_1 \dots$) and $s_1 \in \llbracket \phi \rrbracket_\rho$,
- $\vec{s} \in \llbracket \phi_1 \mathcal{U} \phi_2 \rrbracket_\rho$ if and only if $\exists n. ((s_n \in \llbracket \phi_2 \rrbracket_\rho) \wedge \forall m < n. (s_m \in \llbracket \phi_1 \rrbracket_\rho))$,

It is simple to verify that, for all path-formulas ψ , the set $\llbracket \psi \rrbracket_\rho$ is Borel measurable [1]. Therefore the definition is well specified. Note how the logic **PCTL** can express probabilistic properties, by means of the connectives $\mathbb{P}_{\bowtie q}^\forall$ and $\mathbb{P}_{\bowtie q}^\exists$, as well as (qualitative) properties of the graph underlying the PNTS by means of the quantifiers \forall and \exists .

5.2 The encoding of PCTL

As it turns out, the crucial property of $L\mu$ which allow the encoding of **PCTL** is the possibility of defining the derived threshold modalities of Definition 2.1 which, crucially, are not expressible in the probabilistic μ -calculus ($qL\mu$) of [17, 29, 23, 8]. We reformulate here the result of Proposition 2.2 which straightforwardly extends to the present setting.

Proposition 5.8. *Let (S, \rightarrow) be a PNTS, ϕ a $L\mu$ formula and ρ an interpretation of the variables. Then it holds that:*

$$\llbracket \mathbb{P}_{\bowtie q} \phi \rrbracket_\rho(s) = \begin{cases} 1 & \text{if } \llbracket \phi \rrbracket_\rho(s) \bowtie q \\ 0 & \text{otherwise} \end{cases}$$

The following lemma is also useful.

Lemma 5.9. *Let (S, \rightarrow) be a PNTS, ϕ a $L\mu$ formula and ρ an interpretation of the variables. Then:*

- $\llbracket \mathbb{P}_{>0}(\Diamond X) \rrbracket_\rho(s) = 1$ iff $\exists t. (s \rightsquigarrow t \wedge \rho(X)(t) > 0)$
- $\llbracket \mathbb{P}_{=1}(\Box X) \rrbracket_\rho(s) = 1$ iff $\forall t. (s \rightsquigarrow t \rightarrow \rho(X)(t) = 1)$

Proof. Note that $\llbracket \Diamond X \rrbracket_\rho(s) > 0$ holds if and only if there exists $s \rightarrow d$ such that $\sum_{t \in S} d(t) \rho(X)(t) > 0$ holds. This is the case if and only if $d(t) > 0$ (i.e., $s \rightsquigarrow t$) and $\rho(X)(t) > 0$, for some $t \in S$. The result then follows by Proposition 5.8. The case for $\mathbb{P}_{=1}(\Box X)$ is similar. \square

Remark 5.10. When considering $\{0, 1\}$ -valued interpretations for X , the macro formula $\mathbb{P}_{>0} \Diamond$ expresses the meaning of the diamond modality in classical modal logic with respect to the graph (S, \rightsquigarrow) underlying the PNTS. Similarly, $\mathbb{P}_{=1} \Box$ corresponds to the classical box modality.

We are now ready to define the encoding of **PCTL** into $L\mu$.

Definition 5.11. We define the encoding **E** from **PCTL** formulas to closed $L\mu$ formulas (where $\Box \phi$ stands for the $L\mu$ formula $\Box \phi \sqcap \Diamond \underline{1}$), by induction on the structure of the **PCTL** formulas ϕ as follows:

1. $\mathbf{E}(P) = P$,
2. $\mathbf{E}(\text{true}) = \underline{1}$,
3. $\mathbf{E}(\phi_1 \vee \phi_2) = \mathbf{E}(\phi_1) \sqcup \mathbf{E}(\phi_2)$,

4. $\mathbf{E}(\neg\phi) = \text{negate}(\mathbf{E}(\phi)),$
5. $\mathbf{E}(\exists(\circ\phi)) = \mathbb{P}_{>0}(\Diamond \mathbf{E}(\phi)),$
6. $\mathbf{E}(\forall(\circ\phi)) = \mathbb{P}_{=1}(\Box \mathbf{E}(\phi)),$
7. $\mathbf{E}(\exists(\phi_1 \mathcal{U} \phi_2)) = \mu X. \left(\mathbf{E}(\phi_2) \sqcup (\mathbf{E}(\phi_1) \cap \mathbb{P}_{>0}(\Diamond X)) \right),$
8. $\mathbf{E}(\forall(\phi_1 \mathcal{U} \phi_2)) = \mu X. \left(\mathbf{E}(\phi_2) \sqcup (\mathbf{E}(\phi_1) \cap \mathbb{P}_{=1}(\Box X)) \right),$
9. $\mathbf{E}(\mathbb{P}_{\times q}^\exists(\circ\phi)) = \mathbb{P}_{\times q}(\Diamond \mathbf{E}(\phi)),$
10. $\mathbf{E}(\mathbb{P}_{\times q}^\forall(\circ\phi)) = \mathbb{P}_{\times q}(\Box \mathbf{E}(\phi)),$
11. $\mathbf{E}(\mathbb{P}_{\times q}^\exists(\phi_1 \mathcal{U} \phi_2)) = \mathbb{P}_{\times q} \left(\mu X. \left(\mathbf{E}(\phi_2) \sqcup (\mathbf{E}(\phi_1) \cap \Diamond X) \right) \right),$
12. $\mathbf{E}(\mathbb{P}_{\times q}^\forall(\phi_1 \mathcal{U} \phi_2)) = \mathbb{P}_{\times q} \left(\mu X. \left(\mathbf{E}(\phi_2) \sqcup (\mathbf{E}(\phi_1) \cap \Box X) \right) \right),$

Note that Case 4 is well defined since $\mathbf{E}(\phi)$ is closed by construction.

Remark 5.12. The only occurrences of Łukasiewicz operators $\{\oplus, \odot\}$ and scalar multiplication $(q \phi)$ in encoded **PCTL** formulas appear in the formation of the threshold modalities $\mathbb{P}_{\times q}(_)$. Thus, **PCTL** can be also seen as a fragment of $\text{qL}\mu$ extended with threshold modalities as primitive operations. With the aid of these modalities the encoding is, manifestly, a straightforward adaption of the standard encoding of CTL into the modal μ -calculus (see, e.g., [35]).

The main result of this section establishes the correctness of our encoding. For convenience in the formulation, we identify subsets with their corresponding characteristic functions.

Theorem 5.13. *For every PNTS (S, \rightarrow) , **PCTL**-interpretation $\rho: \text{Prop} \rightarrow (S \rightarrow \{0, 1\})$ of the propositional letters and **PCTL** formula ϕ , the equality $\llbracket \phi \rrbracket_\rho(s) = \llbracket \mathbf{E}(\phi) \rrbracket_\rho(s)$ holds, for all $s \in S$.*

Proof. The proof goes by induction on the structure of ϕ . Cases 1–4 of Definition 5.11 are trivial. Case 5 follows directly from Lemma 5.9. Observing that $\llbracket \Box \phi \rrbracket_\rho(s) = 0$ if $s \not\sim$ and $\llbracket \Box \phi \rrbracket_\rho(s) = \llbracket \Box \phi \rrbracket_\rho(s)$ otherwise, the correctness of Case 6 directly follows from Lemma 5.9. Consider cases 7 and 8. The encoding is of the form $\mu X. (F \sqcup (G \cap H(X)))$, where F and G (by induction hypothesis) and $H(X)$ (by Proposition 5.8) are all $\{0, 1\}$ -valued. Therefore the functor $f \mapsto \llbracket F \sqcup (G \cap H(X)) \rrbracket_{\rho[f/X]}$ maps $\{0, 1\}$ -valued functions to $\{0, 1\}$ -valued functions and has only $\{0, 1\}$ -valued fixed-points. It then follows from Remark 5.10 that the correctness of these two cases can be proved with the standard technique used to prove the correctness of the encoding of CTL into Kozen's μ -calculus (see, e.g., [35]). Consider Case 9. It is immediate to verify that $\bigsqcup_\sigma \{m_\sigma^s(U)\}$, where $U = \llbracket \circ\phi \rrbracket_\rho = \bigcup \{U_{\{s,t\}} \mid t \in \llbracket \phi \rrbracket_\rho\}$, is equal (by induction hypothesis) to $\llbracket \Diamond \mathbf{E}(\phi) \rrbracket_\rho(s)$. The desired equality $\llbracket \mathbb{P}_{\times q}^\exists(\circ\phi) \rrbracket_\rho = \llbracket \mathbb{P}_{\times q} \Diamond \mathbf{E}(\phi) \rrbracket_\rho$ then follows by Proposition 5.8. Case 10 is similar. The two cases 11 and 12 are similar, thus we just consider case 11. Let $\phi = \mathbb{P}_{\times q}^\exists(\psi)$ and $\psi = \phi_1 \mathcal{U} \phi_2$. We denote with Ψ the set of paths $\llbracket \psi \rrbracket_\rho$. Denote by $F(X)$ the formula $\mathbf{E}(\phi_2) \sqcup (\mathbf{E}(\phi_1) \cap \Diamond X)$. By application of Lemma 5.8 it is sufficient to prove that the equality

$$\bigsqcup_\sigma \{m_\sigma^s(\Psi)\} = \llbracket \mu X. F(X) \rrbracket_\rho(s) \quad (15)$$

holds. Note that $\mu X. F(X)$ can be expressed as an equivalent $\text{qL}\mu$ formula by substituting the closed subformulas $\mathbf{E}(\phi_1)$ and $\mathbf{E}(\phi_2)$ with two fresh atomic predicates P_i with interpretations $\rho(P_i) = \llbracket \mathbf{E}(\phi_i) \rrbracket$. The equality can then be proved by simple arguments based on the game-semantics of $\text{qL}\mu$ (see, e.g., [23] and [26]), similar to the ones used to prove that the Kozen's μ -calculus formula $\mu X. (P_2 \vee (P_1 \wedge \Diamond X))$ has the same denotation of the CTL formula $\exists(P_1 \mathcal{U} P_2)$ (see, e.g., [35]). However, to keep the paper self-contained, we provide instead a direct proof of Equation 15 based on denotational semantics.

We first show that the inequality

$$\bigsqcup_\sigma \{m_\sigma^s(\Psi)\} \leq \llbracket \mu X. F(X) \rrbracket_\rho(s) \quad (16)$$

holds. Define $\Psi_k = \{s_0.s_1.s_2 \dots \mid s_0 = s \text{ and } \exists n \leq k. (s_n \in \langle \phi_2 \rangle_\rho \wedge \forall m < n. (s_m \in \langle \phi_1 \rangle_\rho))\}$. Clearly $\Psi = \bigcup_k \Psi_k$. Suppose Inequality 16 does not hold. Then there exists some k and scheduler σ such that

$$m_\sigma^s(\Psi_k) > \llbracket \mu X.F(X) \rrbracket_\rho(s) \quad (17)$$

We prove that this is not possible by induction on k . In the $k = 0$ case, since we are assuming $m_\sigma^s(\Psi_0) > 0$, it holds that $s \in \langle \phi_2 \rangle_\rho$. By inductive hypothesis on ϕ_2 , we know that $\llbracket \mathbf{E}(\phi_2) \rrbracket_\rho(s) = 1$ and this implies that $\llbracket \mu X.F(X) \rrbracket_\rho(s) = 1$, which is a contradiction with Inequality 17. Consider the case $k+1$. Note that if $s \in \langle \phi_2 \rangle_\rho$ then $\llbracket \mu X.F(X) \rrbracket_\rho(s) = 1$ as before, contradicting Inequality 17. So assume $s \notin \langle \phi_2 \rangle_\rho$. Since we are assuming $m_\sigma^s(\Psi_{k+1}) > 0$ it must be the case that $s \in \langle \phi_1 \rangle_\rho$. Similarly, $m_\sigma^s(\Psi_{k+1}) > 0$ and $s \notin \langle \phi_2 \rangle_\rho$ imply that $s \not\rightarrow$ does not hold. This means (see Definition 5.3) that $\sigma(\{s\})$ is defined. Let $d = \sigma(\{s\})$ and observe that $m_\sigma^s(\Psi_{k+1}) = \sum_{t \in S} d(t) m_{\sigma'}^t(\Psi_k)$, where $\sigma'(s_0, s_1, \dots, s_n) = \sigma(s, s_0, s_1, \dots, s_n)$. By induction on k we know that the inequality $m_{\sigma'}^t(\Psi_k) \leq \llbracket \mu X.F(X) \rrbracket_\rho(t)$ holds for every $t \in S$. Thus, by definition of the semantics of \diamond , we obtain $m_\sigma^s(\Psi_k) \leq \llbracket \diamond(\mu X.F(X)) \rrbracket_\rho(s)$. Recall that we previously assumed $s \notin \langle \phi_2 \rangle_\rho$ and $s \in \langle \phi_1 \rangle_\rho$. Hence the equality

$$\llbracket \diamond(\mu X.F(X)) \rrbracket_\rho(s) = \llbracket \mathbf{E}(\phi_2) \sqcup (\mathbf{E}(\phi_1) \sqcap (\diamond \mu X.F(X))) \rrbracket_\rho(s)$$

holds. The formula on the right is just the unfolding $F(\mu X.F(X))$ of $\mu X.F(X)$. This implies the desired contradiction with Inequality 17.

We now prove that also the opposite inequality

$$\bigsqcup_\sigma \{m_\sigma^s(\psi)\} \geq \llbracket \mu X.F(X) \rrbracket_\rho(s) \quad (18)$$

holds. By Knaster-Tarski theorem, $\llbracket \mu X.F(X) \rrbracket_\rho = \bigsqcup_\alpha \llbracket F(X) \rrbracket_{\rho^\alpha}$, where α ranges over the ordinals and $\rho^\alpha = \rho[\bigsqcup_{\beta < \alpha} \llbracket F(X) \rrbracket_{\rho^\beta} / X]$. We prove Inequality 18 by showing, by transfinite induction, that for every ordinal α and $\epsilon > 0$, the inequality

$$\bigsqcup_\sigma \{m_\sigma^s(\psi)\} > \llbracket F(X) \rrbracket_{\rho^\alpha}(s) - \epsilon \quad (19)$$

holds, for all $s \in S$. The case for $\alpha = 0$ is immediate since $\llbracket F(X) \rrbracket_{\rho^0}(s) > 0$ if and only if $\llbracket \mathbf{E}(\phi_2) \rrbracket_\rho(s) = 1$ and this implies $\bigsqcup_\sigma \{m_\sigma^s(\psi)\} = 1$. Consider the case $\alpha = \beta + 1$. If $\llbracket \mathbf{E}(\phi_2) \rrbracket_\rho(s) = 1$ then Inequality 18 holds as above. Thus assume $\llbracket \mathbf{E}(\phi_2) \rrbracket_\rho(s) = 0$. Note that $\llbracket F(X) \rrbracket_{\rho^0}(s) > 0$ only if $\llbracket \mathbf{E}(\phi_1) \rrbracket_\rho(s) = 1$. Thus assume $\llbracket \mathbf{E}(\phi_1) \rrbracket_\rho^\beta(s) = 1$. Under these assumption, $\llbracket F(X) \rrbracket_{\rho^\alpha} = \llbracket \diamond F(X) \rrbracket_{\rho^\beta}$ as it is immediate to verify. By definition of the semantics of \diamond we have:

$$\llbracket \diamond F(X) \rrbracket_{\rho^\beta}(s) = \bigsqcup_{s \rightarrow d} \left(\sum_{t \in S} d(t) \llbracket F(X) \rrbracket_{\rho^\beta}(t) \right)$$

By induction hypothesis on β we know that for every $t \in S$ and $\epsilon > 0$ the inequality $\llbracket F(X) \rrbracket_{\rho^\beta}^\beta(t) < \bigsqcup_\sigma \{m_\sigma^t(\psi)\} + \epsilon$ holds. Therefore the inequality

$$\llbracket \diamond F(X) \rrbracket_{\rho^\beta}(s) < \bigsqcup_{s \rightarrow d} \left(\sum_{t \in S} d(t) \left(\bigsqcup_\sigma \{m_\sigma^t(\psi)\} + \epsilon \right) \right)$$

holds for every $\epsilon > 0$. For each transition $s \rightarrow d$ and scheduler σ define σ^d as $\sigma^d(\{s\}) = d$ and $\sigma^d(s.t_0 \dots) = \sigma(t_0 \dots)$. It follows from unfolding of definitions that that

$$\bigsqcup_{s \rightarrow d} \left(\sum_{t \in S} d(t) \left(\bigsqcup_\sigma \{m_\sigma^t(\psi)\} + \epsilon \right) \right) = \bigsqcup_\sigma \{m_{\sigma^d}^s(\psi)\} + \epsilon$$

and this conclude the proof of Inequality 19 for the case $\alpha = \beta + 1$. Lastly, the case for α a limit ordinal follows straightforwardly from the inductive hypothesis on $\beta < \alpha$.

□

$$\begin{aligned}
t_s^\Gamma(X_i) &= \begin{cases} x_{i,s} & \text{if } (i, s) \in \Gamma \\ \sigma_i x_{i,s}. t_s^{\Gamma \triangleright (i,s)}(\psi_i) & \text{if } (i, s) \notin \Gamma \end{cases} \\
t_s^\Gamma(P) &= \underline{\rho(P)(s)} \\
t_s^\Gamma(\overline{P}) &= \underline{1 - \rho(P)(s)} \\
t_s^\Gamma(q\phi) &= q t_s^\Gamma(\phi) \\
t_s^\Gamma(\phi_1 \bullet \phi_2) &= t_s^\Gamma(\phi_1) \bullet t_s^\Gamma(\phi_2) \quad \bullet \in \{\sqcup, \sqcap, \oplus, \odot\} \\
t_s^\Gamma(\Diamond\phi) &= \bigsqcup_{s \rightarrow d} \bigoplus_{s' \in S} d(s') t_{s'}^\Gamma(\phi) \\
t_s^\Gamma(\Box\phi) &= \bigsqcap_{s \rightarrow d} \bigoplus_{s' \in S} d(s') t_{s'}^\Gamma(\phi) \\
t_s^\Gamma(\sigma_i X_i. \psi_i) &= \sigma_i x_{i,s}. t_s^{\Gamma \cup \{(i,s)\}}(\psi_i)
\end{aligned}$$

Figure 2: Reduction of the Model-Checking problem to Łukasiewicz μ -terms evaluation.

6 Model checking

The *model checking* (or formula evaluation) *problem* of the Łukasiewicz modal μ -calculus is the following: given a closed $L\mu$ formula ϕ , a finite rational transition system (S, \rightarrow) and a state $s \in S$, compute the value $\llbracket \phi \rrbracket(s)$.

In this section we reduce this problem to the evaluation of Łukasiewicz μ -terms. We do this by effectively producing a closed rational μ -term $t_s(\phi)$, with the property that $t_s(\phi) = \llbracket \phi \rrbracket(s)$, whence the rational value of $\llbracket \phi \rrbracket(s)$ can be calculated; for example, by the algorithm in Section 3. The construction of $t_s(\phi)$ is similar in spirit to the reduction of modal μ -calculus model checking to a system of nested boolean fixed-point equations in Section 4 of [22].

We assume, without loss of generality, that all fixed-point operators in ϕ bind distinct variables. Let X_1, \dots, X_m be the variables appearing in ϕ . We write $\sigma_i X_i. \psi_i$ for the unique subformula of ϕ in which X_i is bound. The strict (i.e., irreflexive) *domination* relation $X_i \triangleright X_j$ between variables is defined to mean that $\sigma_j X_j. \psi_j$ occurs as a subformula in ψ_i .

Suppose $|S| = n$. For each $s \in S$, we translate ϕ to a μ -term $t_s(\phi)$ containing at most mn variables $x_{i,s'}$, where $1 \leq i \leq m$ and $s' \in S$. The translation is defined using a more general function t_s^Γ , defined on subformulas of ϕ , where $\Gamma \subseteq \{1, \dots, m\} \times S$ is an auxiliary component keeping track of the states at which variables have previously been encountered. Given Γ and $(i, s) \in \{1, \dots, m\} \times S$, we define:

$$\Gamma \triangleright (i, s) = (\Gamma \cup \{(i, s)\}) \setminus \{(j, s') \in \Gamma \mid X_i \triangleright X_j\}.$$

This operation is used in the definition of Figure 2 to ‘reset’ subordinate fixed-point variables whenever a new variable that dominates them is declared. This is well defined because changing from Γ to $\Gamma \triangleright (i, s)$ or to $\Gamma \cup \{(i, s)\}$ strictly increases the function

$$i \mapsto |\{(i, s) \mid (i, s) \in \Gamma\}|: \{1, \dots, m\} \rightarrow \{0, \dots, n\} \quad (20)$$

under the lexicographic order on functions relative to \triangleright .

For a μ -term t' containing variables $x_{i,s'}$ as above, we write $\text{FVI}(t')$ for the set of pairs in the product set $\{1, \dots, m\} \times S$ that index free variables in t' . The translation defined in Figure 2 clearly satisfies:

Lemma 6.1. *Suppose ϕ' is a subformula of ϕ and $\Gamma \subseteq \{1, \dots, m\} \times S$. Then $\text{FVI}(t_s^\Gamma(\phi')) \subseteq \Gamma$.*

Also, the translation can be defined for any finite PNTS, whether rational or not. When the PNTS is rational, so is the resulting μ -term.

Now consider any $\Gamma \subseteq \{1, \dots, m\} \times S$. By a Γ -*environment*, we mean any function $\gamma: \Gamma \rightarrow [0, 1]$. We use the function γ to generate (the variable part of) an interpretation $\rho^\gamma: \{1, \dots, m\} \rightarrow (S \rightarrow [0, 1])$, as

in Definition 4.4, but with variables **Var** replaced with their indices. This is defined to be the unique interpretation that satisfies the equation below.

$$\rho^\gamma(i) = \begin{cases} \text{lfp} \left(f \mapsto s \mapsto \begin{cases} \gamma(i, s) & \text{if } (i, s) \in \Gamma \\ \llbracket \psi_i \rrbracket_{\rho^\gamma[f/X_i]} & \text{if } (i, s) \notin \Gamma \end{cases} \right) & \text{if } \sigma_i = \mu \\ \text{gfp} \left(f \mapsto s \mapsto \begin{cases} \gamma(i, s) & \text{if } (i, s) \in \Gamma \\ \llbracket \psi_i \rrbracket_{\rho^\gamma[f/X_i]} & \text{if } (i, s) \notin \Gamma \end{cases} \right) & \text{if } \sigma_i = \nu \end{cases}$$

Such a unique interpretation exists because the value of $\llbracket \psi_i \rrbracket_\rho$ depends only on the interpretation of ρ on X_i and variables that strictly dominate it.

Lemma 6.2. *Let ϕ' be a subformula of ϕ . Suppose $\Gamma \subseteq \{1, \dots, m\} \times S$ satisfies the property that, for every $(i, s) \in \Gamma$, the variable X_i is not bound in ϕ' . Let γ be any Γ -environment. Then, for all $s \in S$, it holds that $t_s^\Gamma(\phi')(\gamma) = \llbracket \phi' \rrbracket_{\rho^\gamma}(s)$, where on the right hand side of the equality we are using γ to assign values to the $x_{i,s}$ variables in $t_s^\Gamma(\phi')$ in the obvious way.*

Proof. The lemma is proved by an outer induction on the lexicographic order on the function (20) determined by Γ (as induction hypothesis, we assume the property holds for higher values of this function), and by an inner induction on the structure of ϕ' . We consider the two interesting cases: when ϕ' is a variable, and when it is a fixed-point formula.

We treat first the case in which ϕ' is X_i . If $(i, s) \in \Gamma$ then we simply have:

$$t_s^\Gamma(X_i)(\gamma) = x_{i,s}(\gamma) = \gamma(i, s) = \llbracket X_i \rrbracket_{\rho^\gamma}(s) .$$

If $(i, s) \notin \Gamma$ then $t_s^\Gamma(X_i) = \sigma_i x_{i,s} \cdot t_s^{\Gamma \triangleright (i,s)}(\psi_i)$. Without loss of generality, we consider the case that σ is a least-fixed-point μ . For $v \in [0, 1]$, we define $\gamma \triangleright [v/(i, s)]$ to be the $(\Gamma \triangleright (i, s))$ -environment that assigns v to (i, s) and agrees with γ on $\Gamma \cap (\Gamma \triangleright (i, s))$, and we write γ' for the restriction of γ (equivalently of $\gamma \triangleright [v/(i, s)]$) to $\Gamma \cap (\Gamma \triangleright (i, s))$. Then we have:

$$\begin{aligned} t_s^\Gamma(X_i)(\gamma) &= (\mu x_{i,s} \cdot t_s^{\Gamma \triangleright (i,s)}(\psi_i))(\gamma) \\ &= \text{lfp}(v \mapsto t_s^{\Gamma \triangleright (i,s)}(\psi_i)(\gamma \triangleright [v/(i, s)])) \\ &= \text{lfp}(v \mapsto \llbracket \psi_i \rrbracket_{\rho^{\gamma \triangleright [v/(i, s)]}}(s)) && \text{(by induction hypothesis)} \\ &= \llbracket X_i \rrbracket_{\rho^{\gamma'}}(s) && \text{(by definition of } \rho^{\gamma'}) \\ &= \llbracket X_i \rrbracket_{\rho^\gamma}(s) && \text{(because } \rho^\gamma \text{ and } \rho^{\gamma'} \text{ agree on } X_i) \end{aligned}$$

In the case that ψ' is a least fixed point $\mu_i X_i \cdot \psi_i$, we have

$$\begin{aligned} t_s^\Gamma(\mu X_i \cdot \psi_i)(\gamma) &= (\mu x_{i,s} \cdot t_s^{\Gamma \cup \{(i,s)\}}(\psi_i))(\gamma) \\ &= \text{lfp}(v \mapsto t_s^{\Gamma \cup \{(i,s)\}}(\psi_i)(\gamma \triangleright [v/(i, s)])) \\ &= \text{lfp}(v \mapsto \llbracket \psi_i \rrbracket_{\rho^{\gamma \triangleright [v/(i, s)]}}(s)) && \text{(by induction hypothesis)} \\ &= \llbracket \mu X_i \cdot \psi_i \rrbracket_{\rho^\gamma}(s) . \end{aligned}$$

Here, the last equality follows from the definition of ρ^γ , since $\Gamma \cap (\{i\} \times S) = \emptyset$, due to the assumption relating Γ and bound variables.

The case in which ψ' is a greatest fixed point is analogous. \square

The main result of this section is merely a special case of the previous lemma, relativized to an arbitrary formula ϕ and finite PNTS.

Proposition 6.3. *For any closed $L\mu$ formula ϕ , and state s in a finite PNTS, it holds that $\llbracket \phi \rrbracket(s) = t_s^\emptyset(\phi)$.*

7 Related and future work

The first encodings of probabilistic temporal logics in a probabilistic version of the modal μ -calculus were given in [6], where a version **PCTL**^{*}, tailored to processes exhibiting probabilistic but not nondeterministic choice, was translated into a non-quantitative probabilistic variant of the μ -calculus, which included explicit (probabilistic) path quantifiers but disallowed fixed-point alternation.

In their original paper on quantitative μ -calculi [17], Huth and Kwiatkowska attempted a model checking algorithm for alternation-free formulas in the version of $L\mu$ with \oplus and \odot but without \sqcap , \sqcup and scalar multiplication. Subsequently, several authors have addressed the problem of computing (sometimes approximating) fixed points for monotone functions combining linear (sometimes polynomial) expressions with min and max operations; see [13] for a summary. However, such work has focused on (efficiently) finding outermost (simultaneous) fixed-points for systems of equations whose underlying monotone functions are continuous. The nested fixed points considered in the present paper give rise to the complication of non-continuous functions.

It would be interesting to run experimental comparison of the iterative algorithm for calculating the value of a μ -term against the reduction to linear arithmetic. As mentioned in Section 3.6, we expect the direct algorithm to work better in practice than the non-elementary upper bound on its complexity given by our crude analysis, suggests. Furthermore, as a natural generalization of the approximation approach to computing fixed points, the direct algorithm should be amenable to optimizations such as the simultaneous solution of adjacent fixed points of the same kind, and the reuse of previous approximations when applicable due to monotonicity considerations. It would also be interesting to obtain improved bounds on the complexity of calculating the value of a Łukasiewicz μ -term. Kyriakos Kalorkoti (private communication) has suggested one way of obtaining an exponential upper bound on the run-time. As regards lower bounds, it is easy to show that the problem is at least as hard as the *simple stochastic game* problem [7]. We wonder if computing the value of μ -terms can also be shown to be in $NP \cap co-NP$.

Our results on $L\mu$ are a contribution towards the development of a robust theory of fixed-point probabilistic logics. The beginnings of a systematic study are carried out in [25], where an even richer logic is considered, including the operations of multiplication ($x \times y = xy$) and co-multiplication ($x \otimes y = x + y - xy$). However, this extension has the disadvantage that simple formulas can have irrational (though algebraic) values. For example, consider the term $t = \mu x.(((x \otimes x) \times (x \otimes x)) \sqcup \frac{1}{4})$. The value of this term is the least real number $x \in [0, 1]$ such that $x \geq \frac{1}{4}$ and $x = (2x - x^2)^2$, which is $\frac{3-\sqrt{5}}{2}$. On the positive side, one can adapt the result of Proposition 2.3, using an embedding into the first order theory of real-closed fields, to prove that, although irrational, the values of Łukasiewicz terms extended with (co)multiplications are always algebraic. In a related direction, a fixed-point logic combining Łukasiewicz and (co)multiplication connectives, but based on the Brouwer fixed-point theorem (see discussion in Section 2), is investigated by Spada in [33].

Overall, the Łukasiewicz modal μ -calculus of this paper seems to offer a good balance between expressivity (e.g., it can encode **PCTL**) and algorithmic approachability. In addition, the first author has recently shown in [28] that the process equivalence for probabilistic nondeterministic transition systems characterised by $L\mu$ formulas is the standard notion of *convex bisimilarity* [32]. Thus the quantitative approach to probabilistic μ -calculi may be considered equally suitable as a mechanism for characterising process equivalence as the non-quantitative μ -calculi advocated for this specific purpose in [6] and [10].

One further result of [25] is an interpretation of the Łukasiewicz modal μ -calculus, indeed its extension with (co)multiplication, in terms of a generalized notion of two-player stochastic game. However, although naturally interpreting the meaning of the connectives $\{\sqcup, \sqcap, \times, \otimes\}$, the game semantics given to the connectives of strong disjunction \oplus and conjunction \odot is indirect and intricate. An interesting direction for future research is to design an alternative game semantics that offers a more natural interpretation to the connectives of the Łukasiewicz μ -calculus.

Another problem is to find sound and complete axiomatizations of the Łukasiewicz modal μ -calculus. Such an axiomatization could, for example, constitute a valuable tool for proving, or disproving, a long standing open problem in the literature: the decidability of the satisfiability property for **PCTL** formulas. Axiomatizing the Łukasiewicz modal μ -calculus appears to be a difficult problem as similar results (e.g., axiomatization of Kozen's μ -calculus) required significant efforts to be solved [36]. A complete

axiomatization of Łukasiewicz fuzzy logic is presented in [31]. It might serve as a basis for developing an axiomatization of Łukasiewicz μ -calculus (i.e., without modalities) and, in another direction, of the fixed-point free fragment of Łukasiewicz modal μ -calculus.

Further research will have to explore the relations between quantitative μ -calculi such as $L\mu$ and other frameworks for verification and design of probabilistic systems. Important examples include the *abstract probabilistic automata* of [9], the compositional *assume-guarantee* techniques of [21, 12] and the recent *p-automata* of [18]. In particular, with respect to the latter formalism, we note that the acceptance condition of p-automata is specified in terms of stochastic games whose configurations may have preseeded threshold values whose action closely resembles that of the threshold modalities considered in this work (Definition 2.1). Exploring the relations between p-automata games and $L\mu$ -games [25] could be a fruitful topic for investigation.

Acknowledgements We thank Kousha Etessami, Kyriakos Kalorkoti, Enrico Marchioni, Grant Passmore, Phil Scott, Luca Spada, Colin Stirling and the anonymous reviewers for helpful comments and for pointers to the literature.

References

- [1] C. Baier and J. P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
- [2] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer Berlin Heidelberg, 1995.
- [3] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the AMS*, 21(1), 1989.
- [4] B. Boigelot, S. Jodogne, and P. Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic*, 6(13):614–633, 2005.
- [5] P. Cintula and T. Kroupa. Simple games in Łukasiewicz calculus and their cores. *Kybernetika*, 3:404–419, 2013.
- [6] R. Cleaveland, S. P. Iyer, and M. Narasimha. Probabilistic temporal logics via the modal μ -calculus. In *Proceedings of the Second International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 288 – 305. Springer-Verlag London, 1999.
- [7] A. Condon. The Complexity of Stochastic Games. *Information and Computation*, 96:203–224, 1992.
- [8] L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68:374 – 397, 2004.
- [9] B. Delahaye, J. P. Katoen, K. Larsen, A. Legay, M. Pedersen, F. Sher, and A. Wasowski. Abstract probabilistic automata. *Information and Computation*, 232:66–106, 2013.
- [10] Y. Deng and R. van Glabbeek. Characterising probabilistic processes logically. In *Proceedings of Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 6397 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2010.
- [11] J. Ferrante and C. Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM Journal of Computing*, 4(1):69–76, 1975.
- [12] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Quantitative multi-objective verification for probabilistic systems. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 6605 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2011.

- [13] T. M. Galwitza and H. Seidl. Solving systems of rational equations through strategy iteration. *ACM Transactions on Programming Languages and Systems*, 33(3), 2011.
- [14] B. Gerla. Rational lukasiewicz logic and DMV-algebras. *Neural Networks World*, pages 579–584, 2001.
- [15] B. Gerla. Rational lukasiewicz logic and DMV-algebras. arXiv:1211.5485, Nov 2012.
- [16] P. Hájek. *Metamathematics of Fuzzy Logic*. Springer, 2001.
- [17] M. Huth and M. Kwiatkowska. Quantitative analysis and model checking. In *Proceedings of IEEE Symposium on Logic in Computer Science (LICS)*, pages 111–122. IEEE, 1997.
- [18] M. Huth, N. Piterman, and D. Wagner. p-Automata: New Foundations for discrete-time Probabilistic Verification. *Performance Evaluation*, 69(7), 2012.
- [19] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. *Lecture Notes in Computer Science*, 1119:263–277, 1996.
- [20] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.
- [21] M. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-guarantee verification for probabilistic systems. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 6015 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 2010.
- [22] A. Mader. Modal μ -calculus, model checking and gauß elimination. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1217 of *Lecture Notes in Computer Science*, pages 72–88. Elsevier, 1995.
- [23] A. McIver and C. Morgan. Results on the quantitative μ -calculus $\text{qM}\mu$. *ACM Transactions on Computational Logic*, 8(1), 2007.
- [24] R. McNaughton. A theorem about infinite-valued sentential logic. *The Journal of Symbolic Logic*, 16:1–13, 1951.
- [25] M. Mio. *Game Semantics for Probabilistic μ -Calculi*. PhD thesis, School of Informatics, University of Edinburgh, 2012.
- [26] M. Mio. On the equivalence of denotational and game semantics for the probabilistic μ -calculus. *Logical Methods in Computer Science*, 8(2), 2012.
- [27] M. Mio. Probabilistic Modal μ -Calculus with Independent product. *Logical Methods in Computer Science*, 8(4), 2012.
- [28] M. Mio. Upper-Expectation bisimilarity and lukasiewicz μ -Calculus. In *Proceedings of Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 8412 of *Lecture Notes in Computer Science*, pages 335–350. Elsevier, 2014.
- [29] C. Morgan and A. McIver. A probabilistic temporal calculus based on expectations. In *Proceedings of Formal Methods Pacific*, pages 4–22. Springer-Verlag, 1997.
- [30] D. Mundici. *Advanced Lukasiewicz Calculus and MV-Algebras*. Trends in Logic. Springer-Verlag, 2011.
- [31] A. D. Nola and I. Leustean. Riesz MV-algebras and their logic. In *Proceeding of Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*, pages 140–145. Atlantis Press, 2011.
- [32] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.

- [33] L. Spada. ŁII logic with fixed points. *Archive for Mathematical Logic*, 47:1260–1267, 2008.
- [34] L. Spada. μ MV-algebras: An approach to fixed points in łukasiewicz logic. *Fuzzy Sets and Systems*, 159(10):1260–1267, 2008.
- [35] C. Stirling. *Modal and temporal logics for processes*. Springer, 2001.
- [36] I. Walukiewicz. Completeness of Kozen’s axiomatisation of the propositional μ -calculus. *Information and Computation*, 157, 2000.